HTML5 y CSS3

Revolucione el diseño de sus sitios web (2a edición)

Es necesario que los **diseñadores web**, a la hora de crear sitios web, dominen dos lenguajes fundamentales: el **HTML** (HyperText Markup Language) y el **CSS** (Cascading Style Sheets). Las últimas versiones de estos dos lenguajes, el HTML5 y el CSS3, revolucionarán la forma de crear sitios web ofreciendo nuevas posibilidades de estructura, formato y composición.

Este libro va dirigido a **diseñadores de sitios web** y **grafistas** con conocimientos sobre HTML 4 y CSS 2.1 que quieran **evolucionar en su manera de trabajar** en la creación de sitios que integren esos nuevos estándares.

En el apartado dedicado al **HTML5** se estudiará la **nueva sintaxis**, los nuevos **elementos de estructura** semántica (para el encabezado, las secciones, el menú de navegación, etc.), así como los **formularios interactivos**.

En el apartado de **CSS3**, el lector aprenderá a usar los **nuevos selectores**, el **formato** de **texto** y los estilos para los **contenedores** (sombras, bordes redondeados, degradados, bordes con diseños, etc.)

También podrá descubrir el enorme potencial de los nuevos efectos de **transformación**, **transición** y **animación** y, por supuesto, utilizará una de las grandes novedades: la inserción nativa de **elementos multimedia** (**audio** y **vídeo**).

Se ha dedicado un capítulo al **Diseño web flexible** (Responsive Web Design), que permitirá adaptar los sitios web a los diferentes soportes de difusión (pantallas de ordenador, tabletas táctiles y smartphones).

El último capítulo tratará la aplicación concreta de diseños web a partir del estudio de **plantillas de sitios web** concebidas íntegramente en HTML5 y CSS3.

Christophe AUBRY

Responsable pedagógico en un centro de formación y formador en tecnologías web y artes gráficas durante más de quince años, **Christophe Aubry** dirige en la actualidad la empresa netPlume, especializada en la redacción pedagógica y la creación de sitios web. Autor de multitud de libros publicados en Ediciones ENI, en especial sobre Dreamweaver, WordPress, Drupal, HTML y CSS, se mantiene siempre al tantode las novedades relacionadas con la creación de sitos web, consultando con regularidad las noticias sobre tecnología y participando en numerosos foros.

La creación de sitios web

¿La creación de sitios web es realmente un "arte" hoy en día? Si intenta enumerar las competencias necesarias para concebir un sitio web, puede que que la lista sea impresionante, ya que deberá incluir las habilidades de diversos especialistas: grafista, diseñador, ergonomista, redactor, jefe de proyecto, programador, comercial, community manager, especialista SEO, gestor de servidores, integrador, gestor de la estrategia de los contenidos, director artístico... Y nos paramos aquí, aunque podríamos seguir. Ni que decir tiene que la creación y la gestión de un sitio web es la obra común de diversos profesionales con habilidades complementarias, en la que cada cual será el especialista de su área.

Los lenguajes

La base de todo proyecto web es la página web, esa famosa página que se abrirá en su navegador web. Esta página web estará creada con dos lenguajes fundamentales: el **HTML** (*HyperText Markup Language*) y el **CSS** (*Cascading Style Sheets*).

Estos dos lenguajes han conocido una evolución significativa en su última versión, podríamos hablar incluso de una revolución. ¡Las posibilidades que nos ofrecen son fantásticas! Aunque aún no estén totalmente finalizados, ahora es el turno de los navegadores, quienes deberán aprender a gestionar perfectamente estos nuevos lenguajes.

Vamos a poder crear sitios web atractivos, dinámicos e interactivos usando exclusivamente los lenguajes estándares preconizados por el W3C (*World Wide Web Consortium*).

El libro

1. Para los diseñadores web

Este libro se dirige a **diseñadores web**, y no a programadores web. HTML5 nos trae novedades en cuanto a la concepción y organización de la estructura de los sitios web, además de añadir nuevos elementos para la concepción de aplicaciones web (la geolocalización, el principio de drag and drop, el almacenamiento de datos, la API para los archivos, la Web sin conexión y la Web Sockets API). Esas novedades relacionadas con las aplicaciones (Web Apps) están pensadas para los programadores "puros", y no para los diseñadores web, así que nosotros no veremos aquí esa vertiente de HTML5.

2. El contenido

Vamos a analizar los lenguajes HTML5 y CSS3 con el objetivo de utilizar los elementos esenciales para el diseño web.

En HTML5 veremos la **nueva sintaxis**, los nuevos **elementos de estructura** (`<nav>`, `<header>`...), los nuevos elementos para **los formularios** y presentaremos el elemento de **contenido gráfico**`<canvas>`.

En CSS3 estudiaremos los **nuevos selectores**, el **formateo** del **texto** y de los **contenedores**(sombras, esquinas redondeadas, degradados, diseño de los bordes...).

Aprenderemos a usar los nuevos elementos de **transformación**, de **transición** y de **animación**.

Sin olvidarnos, por supuesto, de la gran novedad, la inserción nativa de **contenido multimedia**:**audio** y **vídeo**.

Otro tema a la moda que también abordaremos: el **diseño adaptativo** (*Responsive Web Design*), que le permitirá adaptar su sitio web a los diferentes soportes de difusión (pantalla de ordenador, tabletas y smartphones).

Por último, terminaremos con el estudio de las **plantillas de sitios web**, totalmente en HTML5/CSS3, donde veremos las aplicaciones concretas de diseño web.

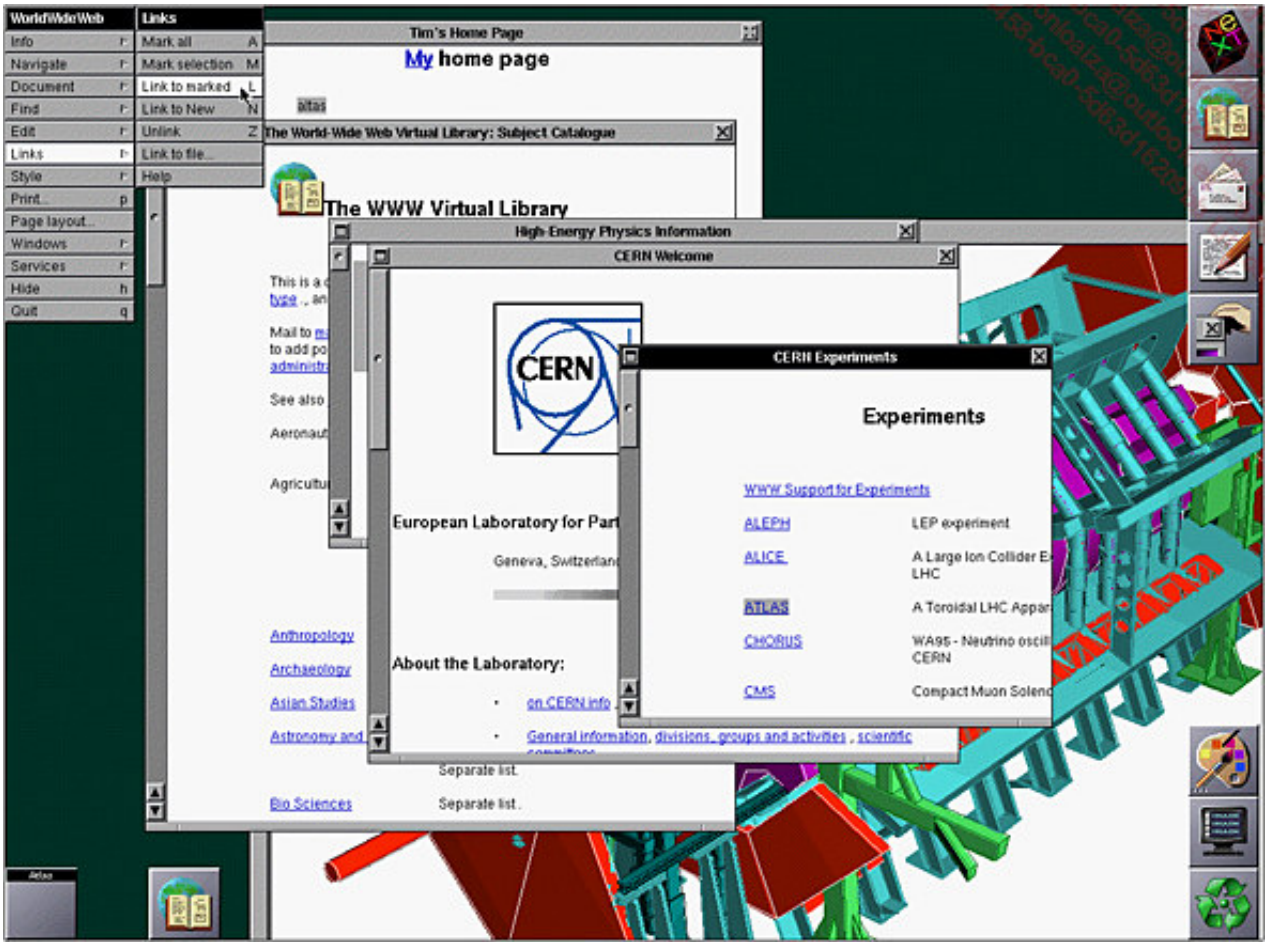
Breve historia de Internet

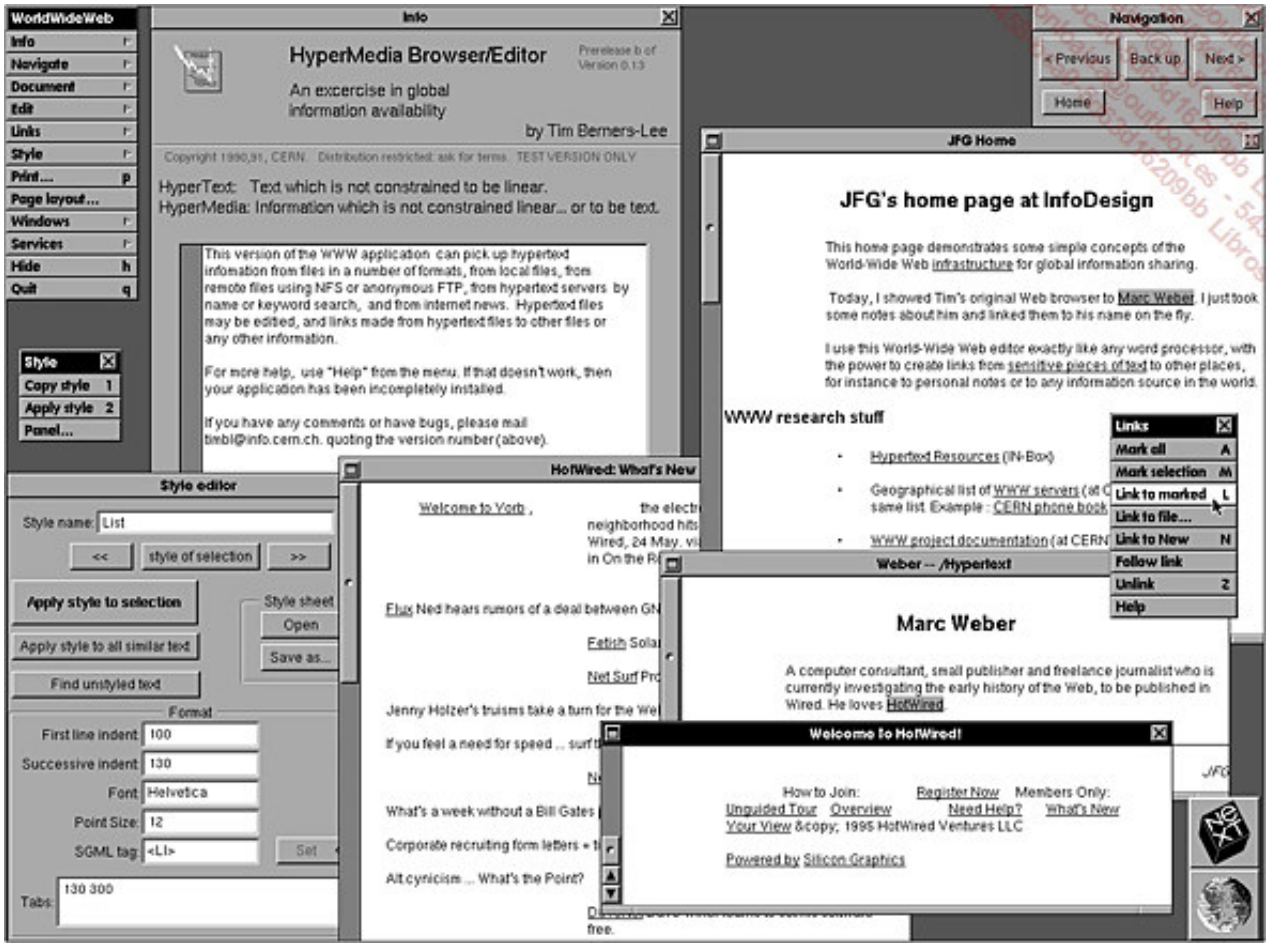
Internet, tal y como nosotros lo conocemos en la actualidad, es un avance de la tecnología bastante reciente. En marzo de **1989**, en la Organización Europea para la Investigación Nuclear (CERN en sus siglas en inglés), Tim Berners-Lee redacta el artículo "fundador" de Internet. En ese artículo titulado "*Information Management: A Proposal*", Tim Berners-Lee habla en su introducción de la gestión de la información mediante un sistema de hipertexto: *"It discusses the problems of loss of information about complex evolving systems and derives a solution based on a distributed hypertext system."*

Esta es la URL de dicho documento: http://www.w3.org/History/1989/proposal.html

En octubre de **1990**, Tim Berners-Lee trabaja sobre el hipertexto con un editor y un navegador en una estación NeXT, y le da a ese programa el nombre de World Wide Web.

Estas son las primeras capturas de pantalla:





En ese mismo mes de octubre de **1990**, el belga Robert Cailliau se une al equipo de Tim Berners-Lee y juntos redactan la segunda propuesta llamada: "*WorldWideWeb: Proposal for a HyperText Project*". Este trabaja sin descanso en la concepción de un navegador para Macintosh.

A finales de **1990** se presenta la primera demostración del primer servidor, del primer editor hipertextual, del primer navegador.

En diciembre de **1992** se instala el primer servidor fuera del CERN, en la universidad de Stanford, en los Estados Unidos.

En el año **1993** aparecen multitud de navegadores y el CERN decide liberar los protocolos web.

Un año más tarde, en **1994**, Mark Andreessen funda Mosaic Communications Corp., futura Netscape.

El **1 de octubre de 1994** nace el Word Wide Web Consortium (W3C) en el MIT (*Massachusetts Institute of Technology*). En ese mismo año 1994, había 623 servidores web en todo el mundo. En abril de 1995 el INRIA (Instituto Nacional francés de Investigación en Informática y Automática) acoge al W3C en Europa y, en septiembre de 1996, lo hace la Universidad de Keio en Japón.

En esta URL podrá acceder a la historia completa: http://www.w3.org/History.html

En el 2004, en el décimo aniversario del W3C, se recoge en una iconografía esta evolución, la historia del World Wide Web. Puede consultarla en esta URL: http://www.w3.org/2005/01/timelines/timeline-2500x998.png

La evolución del HTML

1. Las investigaciones del W3C

El lenguaje HTML es el resultado del trabajo del W3C. Estos trabajos de investigación pasan por varias etapas que han ido evolucionando con el tiempo. En la actualidad, los resultados de las investigaciones se publican en tres etapas:

- Los "borradores", **Working Drafts**, se publican para compartir los avances de las investigaciones con los demás.
- Las **especificaciones** se publican cuando el lenguaje ya está casi terminado y listo para usar.
- Las **recomendaciones** se publican una vez que el lenguaje está oficialmente aprobado y terminado.

A veces lleva mucho tiempo pasar de una especificación a una recomendación, puede que incluso años.

2. El HyperText Markup Language

Fue en 1991 cuando Tim Berners-Lee redactó los primeros "bocetos" (*draft*) del HTML.

En **junio de 1993**, aparece el primer documento técnico describiendo el lenguaje HTML: *"Hypertext Markup Language (HTML) - A Representation of Textual Information and MetaInformation for Retrieval and Interchange"* (http://www.w3.org/MarkUp/draft-ietf-iiir-html-01.txt).

El **8 de noviembre de 1993**, aparece el HTML+ (http://www.w3.org/MarkUp/HTMLPlus/htmlplus_1.html).

La versión 2 del HTML ve la luz el **22 de septiembre de 1995** (http://www.w3.org/MarkUp/html-spec/html-spec_toc.html) bajo los auspicios del IETF (*Internet Engineering Task Force*).

En **marzo de 1995** se publica el HTML 3 (http://www.w3.org/MarkUp/html3/) que se presenta como una "extensión" del HTML 2. Rápidamente es remplazado por la recomendación del HTML 3.2, el **14 de enero de 1997** (http://www.w3.org/TR/REC-html32.html).

La recomendación del HTML 4.01 se publica el **24 de diciembre de 1999**. Para el W3C, esta versión es la última del HTML. EL W3C piensa que el futuro de las páginas web no está en el HTML, sino en el XML. El HTML "ha muerto", para el W3C.

3. La evolución con el XHTML

El lenguaje HTML conlleva limitaciones intrínsecas:

- los elementos que se pueden usar se limitan a los indicados en la recomendación,
- el lenguaje es muy permisivo: es posible indicar el nombre de los elementos en mayúsculas o en minúsculas, algunas etiquetas de cierre son facultativas...
- los elementos no son realmente semánticos: el elemento `<p>` puede contener cualquier tipo de texto.

El W3C publica el **10 de febrero de 1998** (http://www.w3.org/TR/1998/REC-xml-19980210) la recomendación de un nuevo lenguaje, el **XML** (*eXtensible Markup Language*). La quinta edición se remonta al 26 de noviembre de 2008 (http://www.w3.org/TR/2008/REC-xml-20081126/). Este lenguaje permite superar todas las limitaciones del HTML. De este modo es posible crear todos los elementos que queramos, de forma totalmente semántica (por ejemplo, un elemento `<cp>` para contener un código postal o un elemento `<precio>` para incluir un precio) y la sintaxis es muy

estricta. Se trata de un lenguaje ideal, en especial, para intercambiar datos estructurados en la web. Pero el XML interviene además en otros muchos lenguajes estándares del W3C (MathML, SVG...) y en otras tecnologías web.

Debido a su sintaxis permisiva, el HTML es incompatible con el XML. Para resolver esta incompatibilidad, el W3C reformuló la sintaxis del HTML para que fuera compatible con el XML: se trata del **XHTML** (*eXtensible HyperText Markup Language*). La primera recomendación sale a la luz el**26 de enero de 2000** (http://www.w3.org/TR/2000/REC-xhtml1-20000126/).

El XHTML presenta una sintaxis más estricta, como podemos ver en estos dos ejemplos:

Si se abre un elemento, deberá incluirse la etiqueta de cierre:

- sintaxis correcta en HTML: `<p>Mi texto`
- sintaxis correcta en XHTML: `<p>Mi texto</p>`

El HXTML prevé el cierre de los elementos que no dispongan de una etiqueta de cierre:

- sintaxis correcta en HTML: `<br>`
- sintaxis correcta en XHTML: `<br/>`

Además, los elementos XHTML deben escribirse en minúsculas y los valores de los atributos deben estar entrecomillados (con comillas dobles ").

- sintaxis correcta en HTML: `<P ID=intro>`
- sintaxis correcta en XHTML: `<p id="intro">`

El W3C publica la recomendación del XHTML 1.1 el 31 de mayo de 2001 (http://www.w3.org/TR/2001/REC-xhtml11-20010531/). La última edición se remonta a noviembre de 2010 (http://www.w3.org/TR/xhtml11/). Esta versión abordaba el "auténtico XML", sin elementos incorrectos o que no se adaptaran a los estándares. Esto quiere decir que los documentos para la web ya no podían usar el tipo MIME `text/html`.

El XHTML 2 aparece solamente como *Working Draft* en agosto de 2002 (http://www.w3.org/TR/2002/WD-xhtml2-20020805/). Esta debía ser una versión del lenguaje "puro", sin ningún compromiso con el pasado, ¡hasta el punto de que era incompatible con el contenido web existente! Se trató de una iniciativa demasiado radical. El W3C proponía un lenguaje totalmente desconectado de la realidad, hecho para los informáticos y no para los diseñadores web. Al optar por la incompatibilidad con lo existente, el W3C cometió un grave error estratégico.

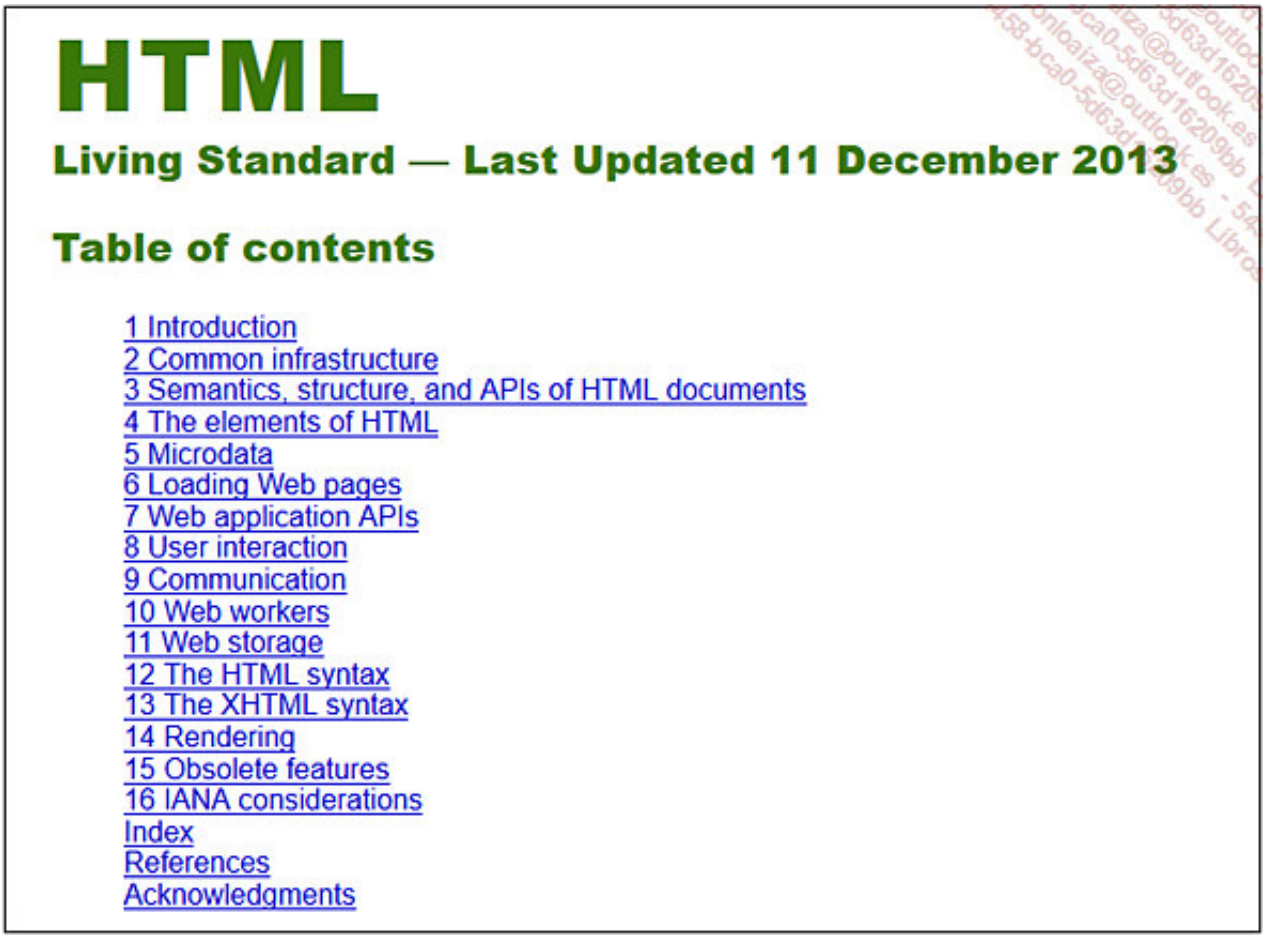
La recomendación del XHTML 2 no llegó nunca a publicarse. El 17 de diciembre de 2010, el W3C disuelve oficialmente ese grupo de trabajo.

4. Los disidentes del WHATWG

Ante ese fracaso de la evolución del HTML reconocido por el W3C, un grupo de "disidentes", encabezado por Ian Hickson (que trabajaba entonces en Opera Software), formó en 2004 su propio grupo de trabajo sobre el HTML, el *Web Hypertext Application Technology Working Group*, WHATWG (http://www.whatwg.org/). Los primeros miembros del **WHATWG** venían de Mozilla, Opera, Apple, y luego, Google. Los primeros resultados de sus investigaciones se aplicaron a los formularios. Ian Hickson era el editor y tomaba todas las decisiones finales. El WHATWG se ocupó en un primer momento de los formularios (Web Forms 2.0) y las aplicaciones web (Web Apps 1.0). Esas dos áreas de trabajo se encuentran ahora en el HTML5.

El WHATWG continúa con esas investigaciones (http://www.whatwg.org/specs/web-apps/current-work/multipage/), independientemente del W3C. Al HTML5 se le llama *HTML Living Standard* para no confundirlo con el HTML5 del W3C.

En julio de 2012, ante las dificultades del trabajo en común y la diferencia de objetivos con respecto al W3C, el WHATWG decide separarse y seguir su propio camino de manera totalmente independiente (véase el documento http://lists.w3.org/Archives/Public/public-whatwg-archive/2012Jul/0119.html). La consecuencia es que, a partir de entonces, el WHATWG propondrá una versión «viva» del HTML y el W3C llevará a cabo la normalización del lenguaje. Para los desarrolladores e integradores Web, eso complicará un tanto la tarea, ya que deberán conocer los elementos HTML5 propuestos por el WHATWG y comprobar si han sido validados por el W3C y son reconocidos por los navegadores.



HTML

Living Standard — Last Updated 11 December 2013

Table of contents

5. La llegada del HTML5

Ante el fracaso del XHTML 2, el W3C forma su propio equipo de trabajo sobre el futuro del HTML, a finales del 2006, tomando como base las investigaciones del WHATWG, en lugar de comenzar desde cero. Una primera especificación en *Working Draft* llamada "*HTML 5 - A vocabulary and associated APIs for HTML and XHTML*" aparece el **22 de enero de 2008** (http://www.w3.org/TR/2008/WD-html5-20080122/). Como anécdota, fíjese que había un espacio entre HTML y 5, que no aparece en la denominación del WHATWG.

En la actualidad (septiembre 2013), la última versión del HTML5 está en **Candidate Recommendation** y tiene fecha de 6 de agosto de 2013 (http://www.w3.org/TR/html5/).

La aparición del logotipo del HTML5 (http://www.w3.org/html/logo/) también provocó un gran revuelo en la web. Algunos lo adoran, otros lo detestan y multitud de parodias gráficas circulan por la red: ¡un auténtico buzz!

6. Los documentos HTML5 del W3C

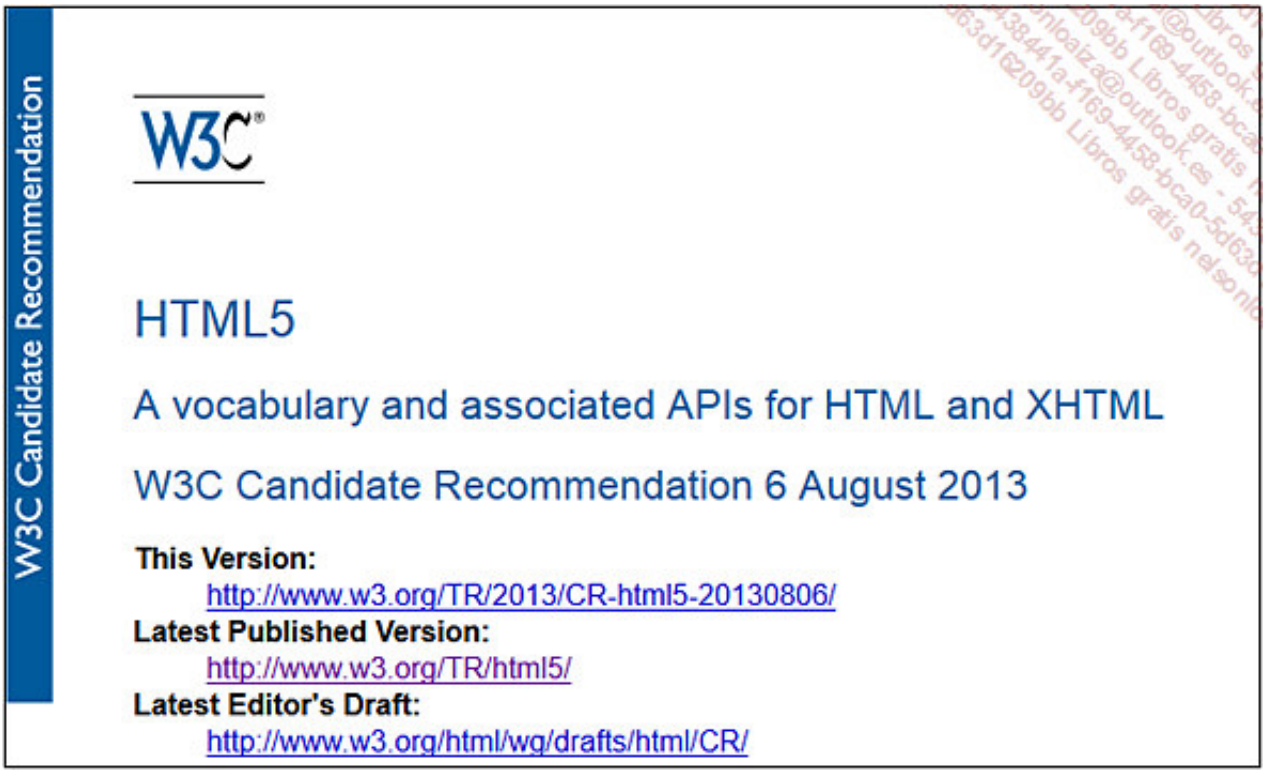
El W3C genera multitud de documentos sobre el HTML5 (y el CCS3).

En primer lugar, veamos cuáles son los objetivos del HTML5:

- Asegurar la compatibilidad con lo que ya existe. Uno de sus principales objetivos es el de no cometer el mismo error que el XHTML 2.
- La especificación describe con detalle lo que deben hacer los navegadores, lo que estos deben mostrar. En el HTML 4.01 había muchos puntos "oscuros", muchas imprecisiones, y cada navegador hacía lo que mejor le parecía. Es necesario documentar lo que ya existe, lo que los navegadores saben hacer desde hace tiempo.
- La especificación regula por fin la gestión de los errores. Si un documento no es correcto, ¿qué deberá mostrar el navegador? Por ejemplo, si nos encontramos con `<strong> <em> texto </strong> </em>`, ¿qué deberá mostrar el navegador?
- Gestionar correctamente la interoperabilidad, la compatibilidad de los navegadores (especificar una gestión idéntica del DOM).

Esos objetivos se recogen en un documento de trabajo: **HTML Design Principles**(http://www.w3.org/TR/html-design-principles/).

El documento de base y el borrador de trabajo, el *Working Draft*, con fecha de 6 de agosto de 2013 (en el momento de redactar estas líneas): http://www.w3.org/TR/html5/. Lleva por título "**HTML5 - A vocabulary and associated APIs for HTML and XHTML**".



Un recuadro "enorme" precisa que el documento no está terminado y que no contiene las últimas evoluciones y correcciones.

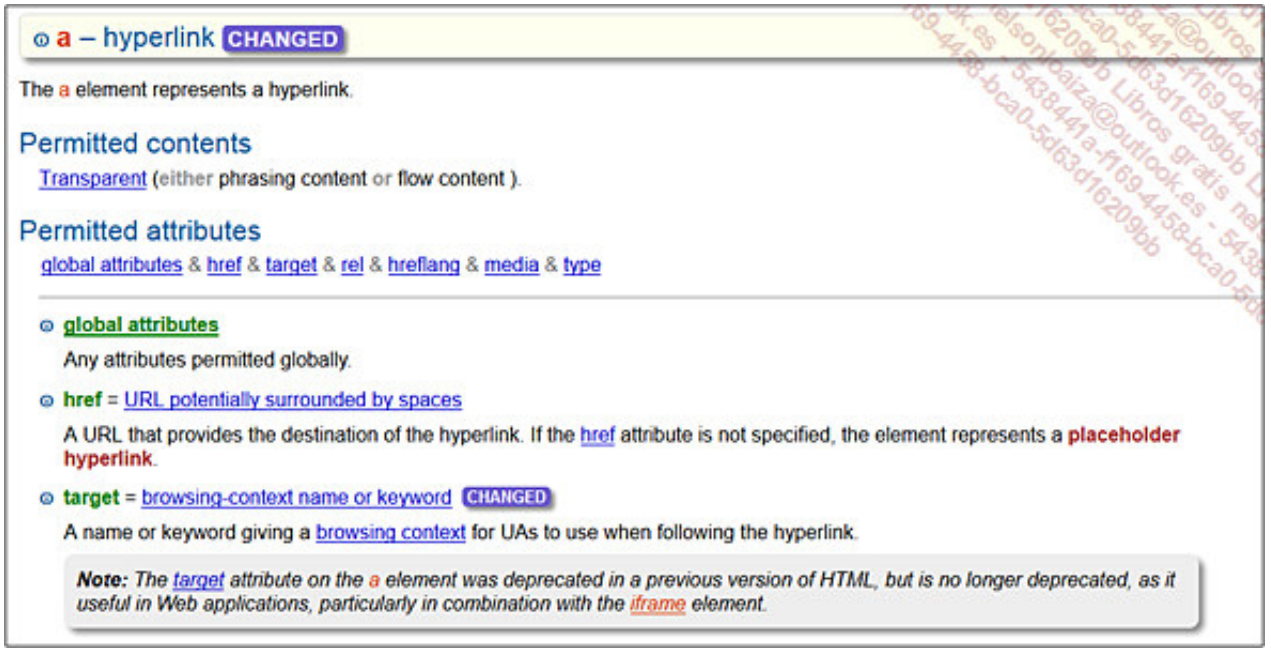
Para conocer los últimos avances del grupo de trabajo (HTML WG), puede consultar la versión "**editor's draft**": http://www.w3.org/html/wg/drafts/html/CR/. La última versión está fechada el 19 de octubre de 2011 (en el momento de redactar este libro).

Otro documento bastante interesante y muy importante para los diseñadores Web, es la referencia HTML (http://dev.w3.org/html5/markup/). Este documento propone únicamente una lista de los elementos HTML utilizables por los diseñadores Web y no incluye toda la parte reservada a los navegadores web (gestión de los errores, interoperabilidad, gestión del DOM, etc.).

Cada elemento HTML aparece con los indicadores **CHANGED** o **NEW**.

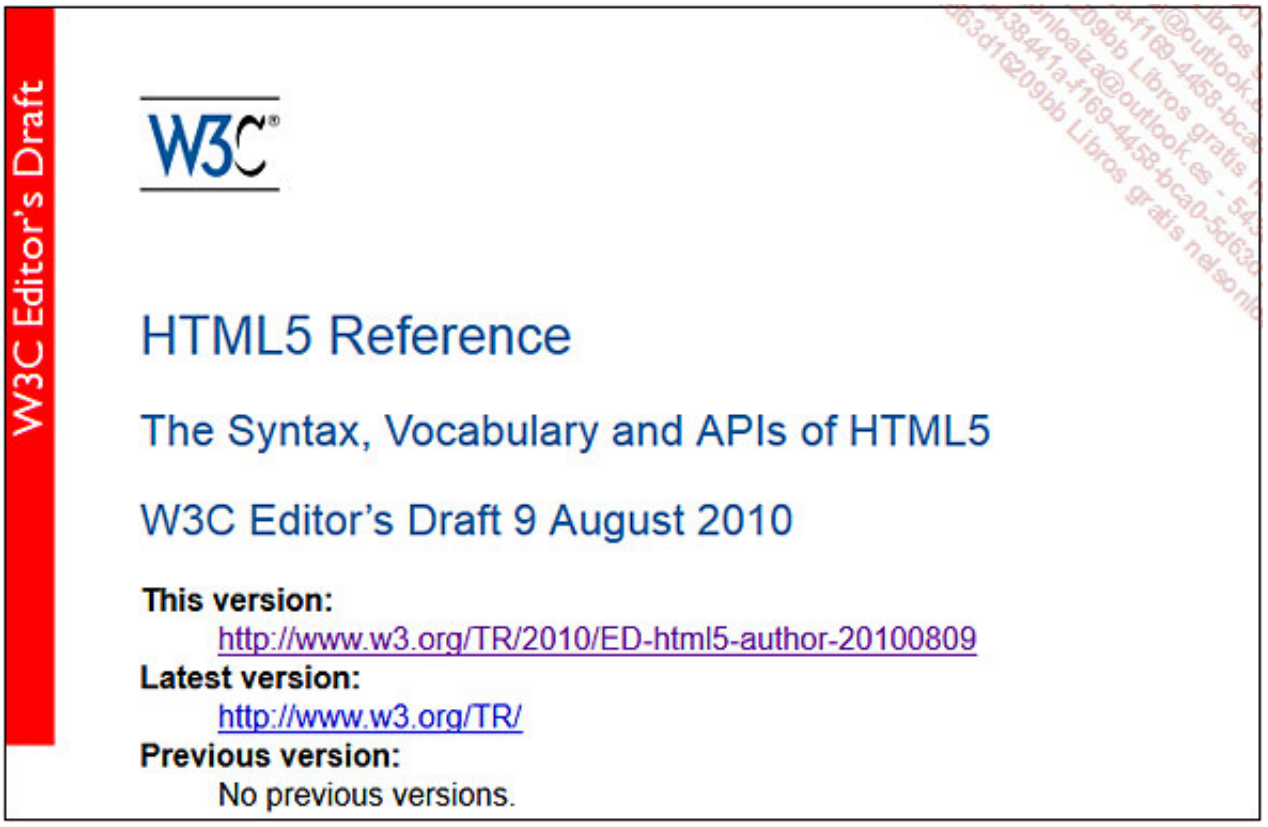
Si hace clic en uno de esos elementos, podrá ver los elementos de sintaxis esenciales para nosotros, los diseñadores web.

Este es un extracto del elemento `a`:



Otro documento interesante es el "*HTML 5 Reference - A Web Developer's Guide to*

HTML 5":http://dev.w3.org/html5/html-author/. Se trata de un documento del 2009, pero contiene multitud de ejemplos de uso de HTML5 para los diseñadores y los autores de sitios web.



7. La evolución de las recomendaciones

El HTML5 pasó a **Candidate Recommendation** el 6 de agosto de 2013 (http://www.w3.org/TR/html5/). Las personas entendidas, como Ian Hickson, predicen que pasará a**Recommendation** ien 2022!

¿Le parece demasiado? Recuerde que el CSS 2.1 pasó a **Recommendation** el 7 de junio de 2011, lo cual no impide que usted lo esté utilizando desde hace diez años.

8. ¿Podemos usar HTML5?

Dado que, en septiembre de 2013, el HTML5 aún no ha alcanzado el estado de **Recommendation**por parte del W3C, ¿podemos utilizarlo sin reparos?

Como veremos en el capítulo La nueva sintaxis HTML5, hoy en día se puede usar HTML5 sin riesgo de incompatibilidad (a no ser que se use un navegador obsoleto como Internet Explorer v6, que data de 2001).

En el capítulo Los elementos de la estructura en HTML5, estudiaremos nuevos elementos de estructura que le permitirán concebir de manera semántica sus páginas web. El riesgo de incompatibilidad aquí es mayor... para los usuarios de Internet Explorer 8. Ya que este último no reconoce y no permite visualizar esos nuevos elementos.

Para los navegadores más antiguos, podemos «forzarlos» a que reconozcan los nuevos elementos HTML5 mediante JavaScript. Con una instrucción JavaScript, solicitaremos crear estos elementos en el DOM (el *Document Object Model* permite acceder a los elementos de estructura de páginas Web).

En el `<head>` de la página Web, introduzca este código:

```
<script type="text/javascript">
```

```
    document.createElement("main");
    document.createElement("header");
    document.createElement("footer");
    document.createElement("section");
    document.createElement("aside");
    document.createElement("nav");
    document.createElement("article");
    document.createElement("figure");
</script>
```

Este script permitirá crear dinámicamente elementos de HTML5 desconocidos por anteriores navegadores. De esta forma podrán mostrar correctamente los nuevos elementos HTML5, sin ignorarlos, tal y como debería ser la regla (véase capítulo Interpretación del HTML5 por los anteriores navegadores).

También es posible crear un script «global» para crear dinámicamente todos los nuevos elementos HTML5. Google propone ese nuevo script: http://code.google.com/p/html5shim/

En el `<head>` de la página Web, introduzca este código:

```
<!--[if lt IE 9]>
<script src="http://html5shim.googlecode.com/svn/trunk/html5.js">
</script>
<![endif]-->
```

El resultado sería idéntico al obtenido con el primer script «manual».

Para los navegadores recientes (posteriores a 2010, como Firefox 3.6, Chrome 4, Opera 10.5 y Safari 5), bastará con gestionar las propiedades `display: block`, para visualizarlos correctamente. De lo contrario se mostrarán como elementos de tipo `inline`.

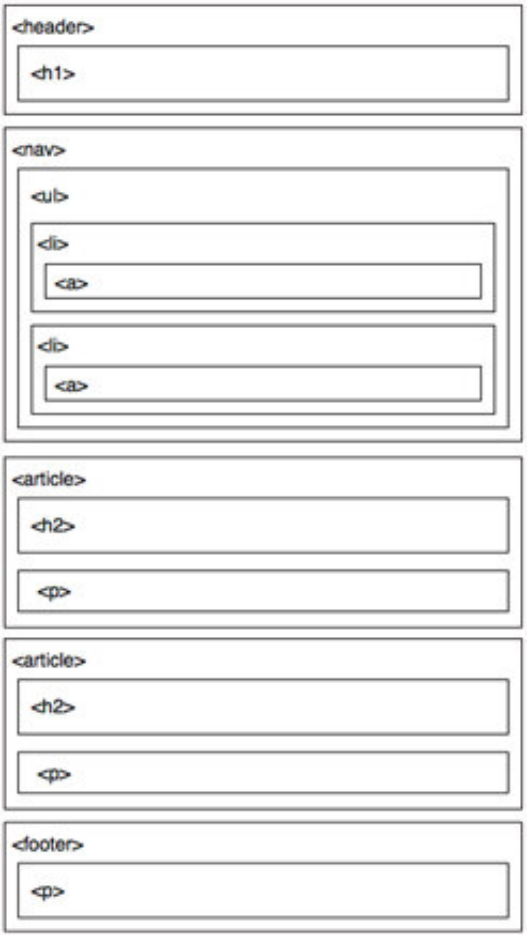
En el capítulo HTML5 y CSS3 para los formularios, veremos los nuevos elementos y los nuevos tipos para los formularios. Los nuevos tipos para los formularios (`email`, `url`, `date`...), cuando no sean reconocidos por el navegador, serán interpretados como objetos de tipo `text`. Nada demasiado grave.

Por último, concluiremos diciendo que: ¡ahora es su turno! Ahora le toca a usted determinar si le conviene o no pasarse a HTML5. Lo que nosotros podemos decir es que los nuevos elementos HTML5 no presentan demasiados riesgos si usted los utiliza con precaución e incluyendo alternativas para los navegadores más antiguos y menos conformes.

9. Interpretación del HTML5 por los anteriores navegadores

No olvide que el HTML5 y las CSS3 también tienen como principio la «tolerancia a fallos». Esto implica que los navegadores deben ignorar todo aquello que no saben interpretar.

Tomemos un ejemplo concreto para comprender bien el proceso. Presentamos a continuación una estructura muy sencilla en HTML5 (estudiaremos estos nuevos elementos en el capítulo Los elementos de la estructura en HTML5).



Internet Explorer 6, por ejemplo, no reconoce estos nuevos elementos HTML5. Por tanto, los ignorará.

En cada uno de esos elementos, añadiremos contenido en los elementos standard HTML 4.1.

Esta es la estructura final del documento de test:

```
<!DOCTYPE html>
<html lang="es">
<head>
   <title>Mi documento en HTML5</title>
   <meta charset="UTF-8" />
</head>
<body>
<header>
   <h1>Mi página Web en HTML5</h1>
</header>
<nav>
   <ul>
      <li><a href="#">Link 1</a></li>
      <li><a href="#">Link 2</a></li>
   </ul>
</nav>
<article>
   <h2>Lunes</h2>
   <p>Donec ullamcorper...</p>
</article>
<article>
   <h2>Martes</h2>
   <p>Morbi leo risus...</p>
</article>
<footer>
   <p>Mi página personal en HTML5</p>
</footer>
```

```
</body>
</html>
```

Añadimos reglas CSS 2.1 para un formato sencillo:

```
<style>
   header, nav, article, footer {
      width: 800px;
      margin: 0 auto;
   }
   header {
      border: 1px solid #333;
      background-color: silver;
   }
   h1 {
      text-align: center;
   }
   nav, article {
      border-bottom: 1px solid #333;
   }
   nav ul {
      margin: 0;
      padding: 0;
   }
   nav ul li {
      display: inline-block;
      margin-right: 20px;
   }
   a {
      text-decoration: none;
      font-weight: bold;
      color: green;
   }
   h2 {
      color: #CC33FF;
   }
   p {
      margin-left: 20px;
   }
   footer {
      background-color: silver;
   }
</style>
```

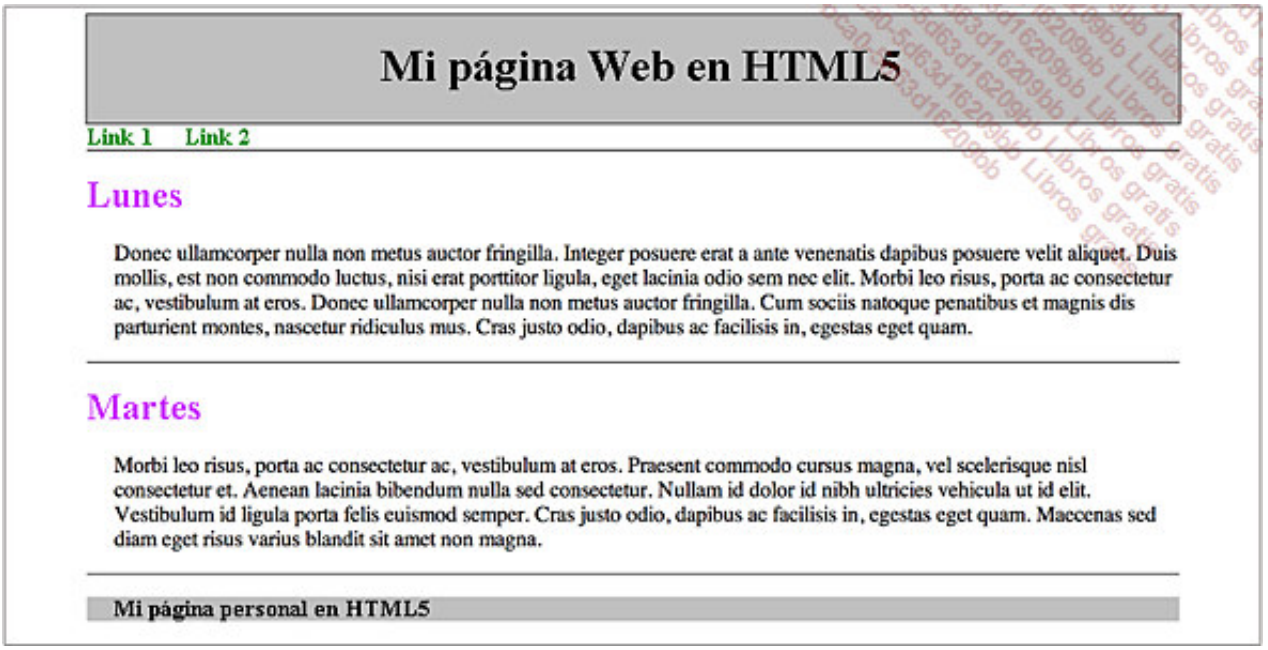
Este formato CSS se aplica a elementos HTML5 (`header`, `nav`...) y HTML 4.1 (`h1`, `h2`, `a`...).

¿Cómo interpretará Internet Explorer 6 esta página?

Explorer ignorará todos los elementos desconocidos (`<header>`, `<nav>`, `<article>`...), ya que no los conoce. Por tanto, si no los conoce, no podrá aplicar su formato y las reglas CSS aplicadas a ellos no se mostrarán.

En cuanto a los elementos conocidos (`<h1>`, `<h2>`, `<p>`...), estos sí se mostrarán, al igual que su formato CSS.

Presentamos cómo se mostraría en un navegador reciente:



Podemos ver todos los elementos.

Así es como se mostraría en Internet Explorer 6:

Internet Explorer 6 no conoce el elemento `<header>` y, por tanto, lo ignora, al igual que su regla CSS.

Internet Explorer 6 conoce el elemento `<h1>` y su formato CSS, entonces lo mostrará correctamente.

Dado que el formato CSS de la lista de vínculos (`<ul>` y `<li>`) declarado en el elemento `nav`, es desconocido para Internet Explorer 6, este no se muestra.

Internet Explorer 6 conoce el color de los vínculos declarado en el elemento `a` y sí lo muestra.

Última especificación: si bien Internet Explorer pasa por alto los elementos desconocidos, ello no quiere decir que los «elimine» del código fuente. Los elementos HTML5 desconocidos estarán presentes en el código fuente, pero no serán interpretados.

10. Este libro y la evolución del HTML5

En definitiva, deberá saber que el trabajo de investigación aún no ha terminado, que se encuentra en evolución permanente y que los cambios son continuos. Esto quiere decir que las especificaciones presentadas en este libro podrían evolucionar: podrían aparecer nuevos elementos, otros elementos podrían definirse de otra manera, o podrían modificarse, y otros elementos podrían declararse obsoletos o incluso podrían desaparecer.

Observe que los borradores de la versión 5.1 están accesibles y datan del 28 de mayo de 2013:http://www.w3.org/TR/html51/

La evolución del CSS

1. La aparición del CSS

Con el HTML 3.2 se daba formato en HTML con los elementos específicos para ello: `<b>`, `<font>`... y sus respectivos atributos `face="arial,helvetica,sans-serif"`, `size="3"`, `align="center"`... Implementarlo requería mucho tiempo, actualizarlo era molesto y no resultaba productivo.

Con la llegada del HTML 4 se hizo "limpieza" de los elementos que se podían usar, pero, sobre todo, llegó acompañado de la hojas de estilo en cascada, *Cascading Style Sheet* o **CSS**. La primera recomendación data del **17 de diciembre de 1996**: http://www.w3.org/TR/REC-CSS1-961217.html

2. La utilidad del CSS

Las hojas de estilo CSS permiten:

- separar la estructura de las páginas y su contenido (HTML), del formato del texto y de la página,
- tener muchas más posibilidades de formato y de presentación de la página,
- definir un estilo una sola vez y poder aplicarlo tantas veces como se desee,
- evitar los errores causados por la repetición,
- reagrupar todos los estilos,
- hacer actualizaciones de forma extremadamente rápida.

En resumidas cuentas, todo son ventajas.

3. Las versiones del CSS

La versión "**level 1**" data, como dijimos, de diciembre de 1996. La versión "**level 2**" data del 12 de mayo de 1998. Se trataba de una versión muy ambiciosa, ¡demasiado ambiciosa incluso! La gran cantidad de novedades (fuentes, síntesis vocal...) no fueron adoptadas por los navegadores. Ante ese "fracaso parcial" el W3C volvió a retomar el trabajo para publicar, en el período de 2004-2006, la versión **level 2 revision 1**, que se conoce como **CSS 2.1**. La versión 2.1 de las CSS fue publicada como **Recommendation** el 7 de junio de 2011.

4. La llegada del CSS3

Los primeros borradores del CSS3 ven la luz en 1999. Esta vez, ante el importante volumen de trabajo, el W3C no publicó una recomendación "descomunal", sino una veintena de "módulos" independientes los unos de los otros. Esto resulta bastante práctico para los navegadores, que pueden así implementar las novedades progresivamente.

En esta URL podrá mantenerse informado de las novedades sobre el CSS3:http://www.w3.org/Style/CSS/current-work. Volveremos a hablar de esto con más detalle en el capítulo El camino hacia el CSS3".

Las principales novedades de CSS3 son las posibilidades de:

- aplicar imágenes a los bordes y añadir varios bordes,
- crear fondos con degradados y con imágenes múltiples,

- usar la transparencia con los colores y con los elementos,
- aplicar sombras a los elementos (cajas, texto, etc.),
- aplicar transformaciones, transiciones y animaciones a las propiedades y a los elementos,
- insertar fuentes con caracteres diversos,
- crear sitios web que se adapten a los distintos tamaños de pantalla.

5. La llegada del CSS4

El CSS 2.1 aún no tiene una recomendación oficial, el CSS3 está en plena transformación... aun así, el W3C ya ha empezado a trabajar en el CSS4. El último borrador (*Working Draft*) sobre los selectores CSS4 se publicó el 2 de mayo de 2013: http://www.w3.org/TR/selectors4/

El doctype

1. La sintaxis en HTML 4 y XHTML 1.0

La primera línea de todo documento HTML corresponde a la declaración del tipo de documento, el**doctype**. Esta **DTD** (*Document Type Declaration*) sirve para indicar qué versión del lenguaje HTML se ha utilizado en el documento.

Esta sería la DTD de un documento HTML 4.01 transicional:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

Y esta sería la DTD de un documento XHTML 1.0 estricto:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

En esta declaración se indica el nombre de la DTD y se facilita la URL en la que los navegadores podrán acceder a la fuente original de dicha declaración. Su sintaxis no siempre resulta comprensible para todo el mundo, aprehenderla requiere tiempo y esfuerzo. Además, seamos realistas, todos nosotros, y yo el primero, siempre nos hemos limitado a copiar y pegar esta línea de un proyecto a otro.

2. La sintaxis en HTML5

La declaración de la DTD en HTML5 no puede ser más sencilla:

```
<!DOCTYPE html>
```

Lo más sorprendente, además de la brevedad de esta sintaxis, es la ausencia del número de versión. Ahora bien, podríamos preguntarnos lo siguiente: *"¿Cómo sabrá el navegador qué versión de HTML hemos usado en nuestro documento?*".

No debemos olvidarnos de que uno de los objetivos del HTML5 es el de dar soporte a las versiones anteriores y, también, a las futuras versiones del lenguaje. Precisamente, por ese motivo, indicar un número de versión ha quedado obsoleto, incluirlo no tendría absolutamente ninguna utilidad.

Recordemos qué función tiene exactamente la DTD: esta declaración no está destinada en absoluto a los navegadores, quienes, pase lo que pase, interpretarán y mostrarán sus páginas web, independientemente de la versión de HTML que haya (o no) indicado. La DTD se dirige exclusivamente a los motores de validación HTML/CSS.

No debemos olvidarnos de que es posible ver la validación como una forma de redactar un documento web semántico, y no como un objetivo en sí mismo. Los navegadores mostrarán siempre un documento web, aun cuando este no sea totalmente válido.

En estas tres URL podrá validar la sintaxis de sus páginas web en HTML5:

- http://validator.w3.org/
- http://jigsaw.w3.org/css-validator/
- http://HTML5.validator.nu/

El documento HTML

1. La sintaxis en XHTML 1.0

El elemento <html> indica, en una página web, el inicio del contenido de la página HTML. Esta sería la sintaxis completa en XHTML 1.0 estricto:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es" >
```

Nuevamente, la brevedad no es aquí la prioridad.

2. La sintaxis en HTML5

En HTML5 lo que prima es la simplicidad, solo se conserva lo esencial. Esta sería la sintaxis en HTML5:

```
<html lang="es" >
```

Bastará con precisar el idioma de la página. Eso es todo.

Observe que es perfectamente posible omitir el elemento <html>, la sintaxis será válida.

La codificación de los caracteres

1. La sintaxis en HTML 4 y XHTML 1.0

Se suele indicar qué codificación de caracteres se ha usado en una página web. Antes se indicaba dos veces: en un elemento `meta`, en la cabecera http, y en el contenido, con el atributo `charset`.

En HTML 4.01 transicional:

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

En XHTML 1.0 estricto:

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
```

2. La sintaxis en HTML5

Una vez más, en HTML5 se le ha dado prioridad a la simplicidad:

```
<meta charset="UTF-8" />
```

Disponemos así de lo estrictamente necesario para una correcta gestión por parte de los navegadores web.

Los scripts

La declaración de los scripts también ha perdido importancia y la sintaxis es ahora más concisa. Se ha pasado de:

```
<script src="miScript.js" type="text/javascript"></script>
```

a:

```
<script src="miScript.js"></script>
```

Esta simplificación se debe a que ahora se da por sentado que los scripts están escritos en JavaScript. Por otro lado, ¿qué otro lenguaje se podría usar, si no?

Los estilos CSS

Partiendo siempre del mismo principio: simplificación y unificación del lenguaje en la declaración de los estilos CSS, se ha pasado de:

```
<link href="miHoja.css" rel="stylesheet" type="text/css" />
```

a:

```
<link href="miHoja.css" rel="stylesheet" />
```

La sintaxis de los elementos

1. Objetivos

Los objetivos que persigue la sintaxis de los elementos HTML son la simplicidad (¡una vez más!) y la permisividad. El lenguaje HTML5 no es una evolución del XHTML, HTML5 no tiene nada que ver con el XML. Por ese motivo ya no tenemos las reglas estrictas de sintaxis del XHTML.

2. Las comillas

Las comillas son facultativas para los valores de los atributos. Por ejemplo, para la codificación de los caracteres podemos usar las tres sintaxis siguientes:

```
<meta charset=UTF-8>
<meta charset='UTF-8'>
<meta charset="UTF-8">
```

Con comillas simples, con comillas dobles o sin comillas, en los tres casos la sintaxis será válida.

En cambio, si un atributo tiene varios valores, será obligatorio el uso de comillas dobles:

```
<div class="estilo1 estilo2">
```

3. Los elementos con una única etiqueta de apertura

Algunos elementos HTML disponen de una única etiqueta de apertura y no tienen etiqueta de cierre: `img`, `br`, etc. En esos casos será necesario indicar el cierre del elemento en la etiqueta de apertura usando `/`.

Ejemplos:

```
<br />
<img src="imagen.png" />
```

En HTML5 no es obligatorio indicar el cierre de esas etiquetas.

Ejemplos válidos:

```
<br>
<img src="imagen.png">
```

4. Los elementos con etiqueta de cierre facultativa

En HTML 4 no es obligatorio incluir la etiqueta de cierre de ciertos elementos. Citemos algunos: `<p>`, `<dl>`, `<td>`... En XHTML, cada vez que se abre una etiqueta es obligatorio cerrarla. HTML5 retoma la sintaxis permisiva del HTML 4. Así, el siguiente ejemplo sería válido en HTML5: `<p>Mi párrafo sin etiqueta de cierre`.

5. Las mayúsculas y minúsculas

HTML5 acepta sin problemas las mayúsculas y las minúsculas en la sintaxis de los elementos y de los atributos. Los siguientes ejemplos son totalmente válidos en HTML5:

```
<SCRIPT src="miScript.js"></script>
<script SRC="MIscript.JS"></SCRIPT>
<sCriPt SrC="MiScript.js"></scRIpt>
```

Es cierto que vistos así, ¡esos ejemplos dan ganas de echarse a correr! Más adelante veremos que, a pesar de todo, es preferible respetar ciertas normas de uso.

6. Los atributos boleanos

Algunos atributos solamente admiten valores boleanos a la hora de aplicar dicho valor.

Este sería el ejemplo de una casilla que aparece seleccionada desde que se abre la página web. Se trata del uso del valor `checked` para el atributo `checked`.

```
<input type="checkbox" checked="checked" name="checkedbox">
```

Resulta redundante indicar ambos parámetros. En HTML5, la simple presencia del atributo es suficiente:

```
<input type="checkbox" checked name="checkedbox">
```

Nuevamente, se ha dado prioridad a la concisión de la sintaxis.

7. Los elementos HTML, HEAD y BODY

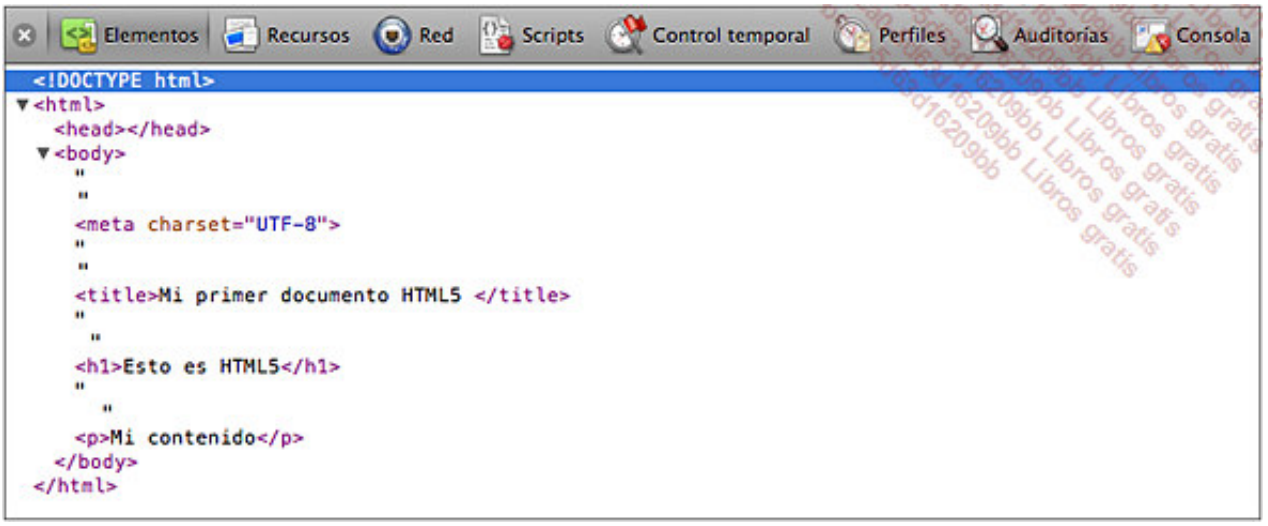
En cuanto a concisión se refiere, el HTML5 ha hecho un gran esfuerzo, colosal incluso: los elementos`html`, `head` y `body` son ahora facultativos.

Este sería un documento HTML totalmente válido y que se visualizaría correctamente:

```
<!DOCTYPE HTML>
<meta charset="UTF-8" />
<title>Mi primer documento HTML5 </title>
<h1>Esto es HTML5</h1>
<p>Mi contenido</p>
```

Se visualizará así:

Si examina el código fuente, podrá ver que los navegadores añaden ellos mismos los elementos HTML que faltan.



Como puede ver, los elementos `<thead>`, `<tfoot>` y `<tbody>` también son facultativos.

8. La sintaxis recomendada

Acabamos de ver que el HTML5 es muy permisivo. Podemos escribir nuestro código como prefiramos, casi podríamos decir que ide cualquier manera! La ventaja de esto es que ofrece una gran flexibilidad, con muy pocas restricciones. La principal desventaja reside en que las páginas así redactadas no se parecerán a nada de lo que ya conocemos, de modo que cada diseñador web deberá analizar durante "cierto tiempo" las páginas HTML de un proyecto "mal redactado" en las que deba trabajar, antes de comprender lo que tiene entre manos. Al ser demasiado permisivos, nos exponemos a perder el lenguaje común, que todos conocemos y usamos.

Por ese motivo le recomiendo que **respete la sintaxis del XHTML**. Se trata de una sintaxis que todos conocemos, ha sido adoptada por todos los creadores de sitios web, es la misma para todos y forma parte de las "prácticas recomendadas" para los diseñadores web.

Respetemos las normas de uso y de sintaxis del XHTML:

- usar solamente minúsculas,
- usar siempre las comillas,
- cerrar con / las etiquetas que se auto-cierran,
- cerrar los elementos para los cuales la etiqueta de cierre sea facultativa,
- sangrar el código para aumentar la legibilidad.

Los elementos obsoletos

1. ¿Qué es un elemento obsoleto?

En la especificación del HTML5, determinados elementos se consideran "obsoletos". Un elemento obsoleto es un elemento que no ha sido declarado en la especificación del HTML5. Sin embargo, como el HTML5 debe ser compatible con las versiones anteriores, los elementos HTML obsoletos seguirán siendo interpretados correctamente por los navegadores.

Una página web que use elementos HTML obsoletos se visualizará correctamente, aunque los motores de validación la consideren como "no conforme".

Esta es la URL de los elementos obsoletos del sitio web del W3C:http://www.w3.org/TR/html5/obsolete.html

2. Para una mejor accesibilidad

Los marcos, antiguo elemento de concepción de páginas web, se han declarado obsoletos, para alegría de la accesibilidad de los sitios web y del posicionamiento en buscadores.

Los elementos HTML `<frame>`, `<frameset>` y `<noframes>` no se deben usar.

3. Los elementos en desuso o mal utilizados

Los elementos `<applet>`, `<isindex>` y `<dir>` se han declarado obsoletos.

Los elementos redefinidos

En HTML5 se ha redefinido la función de determinados elementos HTML, en lugar de declararlos obsoletos. Esos elementos están relacionados con el texto, así que los veremos en el capítulo dedicado al texto.

Los atributos obsoletos

La lista de atributos obsoletos es larga. Los atributos que se consideran obsoletos han sido remplazados por el CSS, que es más eficaz. Para consultar la lista completa de los elementos obsoletos, acceda a la siguiente URL del sitio web del W3C:http://www.w3.org/TR/html5/obsolete.html

Nuevos elementos

HTML5 introduce algunos elementos nuevos. No debemos olvidarnos de que la especificación del HTML5 se encuentra en continua evolución, por lo que es posible que aparezcan otros elementos nuevos tras la redacción de este libro. Esta lista no es exhaustiva.

1. El elemento <hgroup>

En HTML 4, si usted quería incluir un título y un subtítulo en una página web, podía utilizar esta sintaxis:

```
<h1>Mi sitio web</h1>
<p>Un sitio web sobre la creación de sitios web</p>
```

En este ejemplo, nada indica que exista un subtítulo.

También podría haber usado esta sintaxis:

```
<h1>Mi sitio web</h1>
<h2>Un sitio web sobre la creación de sitios web</h2>
```

En ese caso, el resto de títulos de la página deberían usar las etiquetas de título que van de <h3>a <h6>.

Con HTML5, el elemento <hgroup> permite agrupar los elementos del encabezado de tipo <h1> a<h6>. Resulta práctico para incluir en una página un título y un subtítulo, perfectamente definidos desde un punto de vista semántico.

Esta sería la sintaxis que debería usarse:

```
<hgroup>
  <h1>Mi sitio web</h1>
  <h2>Un sitio web sobre la creación de sitios web</h2>
</hgroup>
```

Los elementos <hx> dentro de <hgroup> no tienen que empezar necesariamente con <h1>, sino que pueden tener el nivel de título que usted prefiera. En otras palabras, no es obligatorio seguir una jerarquía estricta a la hora de usar las etiquetas <hx>. Se trata simplemente de una lista de elementos <hx> que se pueden combinar unos con otros. El objetivo de <hgroup> no es el de crear un índice.

Atención, este elemento, propuesto en otro tiempo por el W3C, se considera en la actualidad obsoleto (véase la última lista de elementos obsoletos con fecha de 6 de agosto de 2013:http://www.w3.org/TR/html5/obsolete.html).

2. El elemento <time>

Este elemento permite indicar que su contenido está relacionado con el factor tiempo: fecha y/o hora. Simplemente aporta un valor semántico, no se mostrará automáticamente la fecha o la hora.

Estos serían algunos ejemplos de su sintaxis:

```
<p>La fecha de hoy es: <time>05/09/2013</time></p>
<p>Hoy es <time>5 de septiembre</time>.</p>
```

Puede usar el atributo datetime para indicar la fecha o la hora en formato ISO 8601

(http://es.wikipedia.org/wiki/ISO_8601):

```
<p>La fecha de hoy es: <time datetime="2013-09-05">5
de septiembre</time>.</p>
```

También puede utilizar el atributo `pubdate`. Este atributo permite especificar que la fecha indicada es aquella de la publicación del ancestro más cercano del elemento `<article>`.

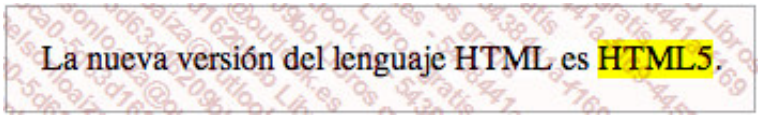
3. El elemento <mark>

Este elemento permite resaltar un texto dentro de un texto más general.

Veamos un ejemplo:

```
<p>La nueva versión del lenguaje HTML es <mark>HTML5</mark>.</p>
```

El texto aparecerá resaltado por lo general con un fondo amarillo.



4. El elemento <meter>

Este elemento permite definir una medida determinada dentro de un contexto de valores específicos. Dicho elemento utiliza seis atributos:

- `value`: el valor del dato en la escala utilizada.
- `min`: el mínimo posible.
- `low`: el mínimo alcanzado.
- `max`: el máximo posible.
- `hight`: el máximo alcanzado.
- `optimum`: el valor mínimo ideal.

Veamos un ejemplo:

```
<p>Usted ha obtenido la nota de 8 sobre 10: <meter value="8" min="0" low="4"
max="10" hight="8" optimum="9"></p>
```

Cada navegador podrá mostrarlo como prefiera. Esta sería la visualización con Google Chrome:

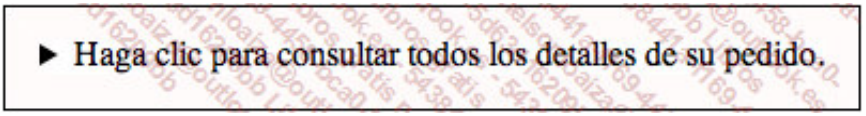
5. Los elementos <details> y <summary>

El elemento `<summary>` permite presentar un resumen de la información que se facilitará con el elemento `<details>`.

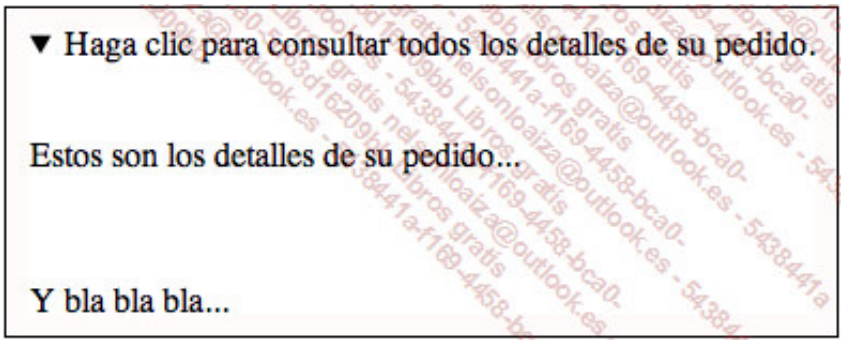
Ejemplo:

```
<details>
  <summary>Haga clic para consultar todos los detalles de su pedido.</summary>
  <p>Estos son los detalles de su pedido...</p>
  <p>Y bla bla bla...</p>
</details>
```

Cada navegador podrá decidir cómo desea mostrar esos detalles. Esta sería la visualización con Google Chrome:



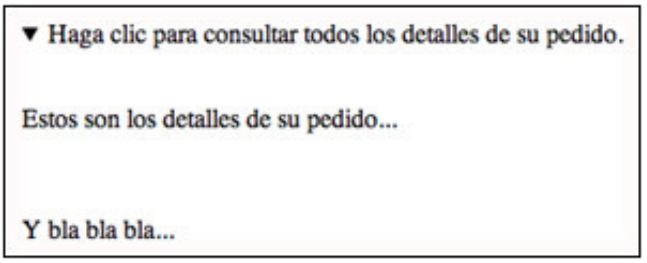
Si hace clic en el triangulito de la izquierda podrá acceder a los detalles:



Puede utilizar el atributo `open` en el elemento `<details>`. Esto permitirá indicar si los detalles deben abrirse -es decir, estar visibles-, cuando se cargue la página.

```
<details open="open">
   <summary>Haga clic para consultar todos los detalles de su pedido</summary>
   <p>Estos son los detalles de su pedido...</p>
   <p>Y bla bla bla...</p>
</details>
```

Este es el resultado:



6. El elemento <wbr>

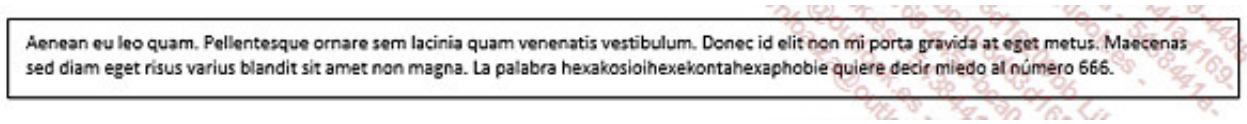
El elemento `<wbr>` (*word break*) permite proponer cesuras dentro de las palabras o frases. Esto ayudará a que los motores de transcripción efectúen las cesuras en el lugar que nos parezca más apropiado. Observe que no crean una pausa, no posicionan un guion, simplemente proponen a los motores de transcripción un lugar por donde cortar. Nosotros deberemos situar el elemento allí donde deseemos proponer una pausa adecuada.

Ejemplo:

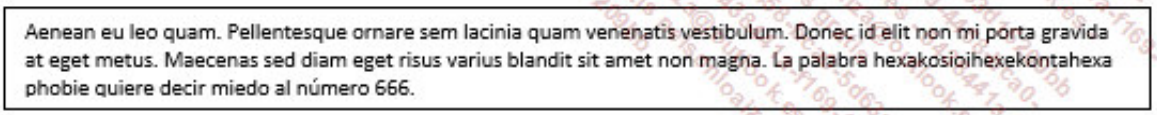
```
<p>Aenean eu leo quam. Pellentesque ornare sem lacinia quam
venenatis vestibulum. Donec id elit non mi porta gravida at eget
```

```
metus. Maecenas sed diam eget risus varius blandit sit amet non
magna. La palabra hexa<wbr>kosioi<wbr>hexekonta<wbr>hexa<wbr>phobie
quiere decir miedo al número 666</p>
```

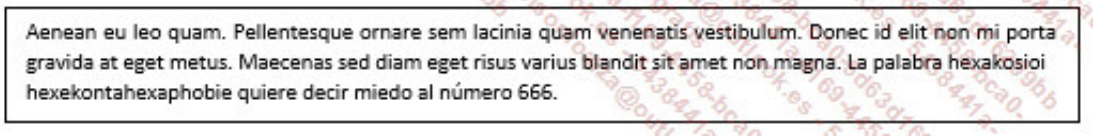
Vista con una anchura considerable donde las cesuras no intervienen:


Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis vestibulum. Donec id elit non mi porta gravida at eget metus. Maecenas sed diam eget risus varius blandit sit amet non magna. La palabra hexakosioihexekontahexaphobie quiere decir miedo al número 666.

Vista con una anchura reducida donde intervienen las cesuras:


Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis vestibulum. Donec id elit non mi porta gravida at eget metus. Maecenas sed diam eget risus varius blandit sit amet non magna. La palabra hexakosioihexekontahexa phobie quiere decir miedo al número 666.

Segundo ejemplo de cesura:


Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis vestibulum. Donec id elit non mi porta gravida at eget metus. Maecenas sed diam eget risus varius blandit sit amet non magna. La palabra hexakosioi hexekontahexaphobie quiere decir miedo al número 666.

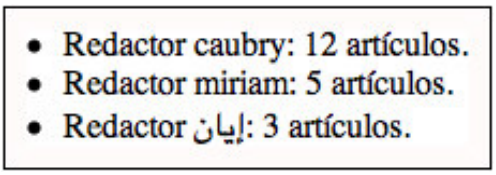
7. El elemento <bdi>

El elemento `<bdi>` permite aislar un texto que tenga una dirección de lectura diferente, como el árabe, que se lee de derecha a izquierda.

Ejemplo:

```
<ul>
  <li>Redactor <bdi>caubry</bdi>: 12 artículos.</li>
  <li>Redactor <bdi>miriam</bdi>: 5 artículos.</li>
  <li>Redactor <bdi>إيان</bdi>: 3 artículos.</li>
</ul>
```

Que se visualizaría así:


- Redactor caubry: 12 artículos.
- Redactor miriam: 5 artículos.
- Redactor إيان: 3 artículos.

8. El elemento <progress>

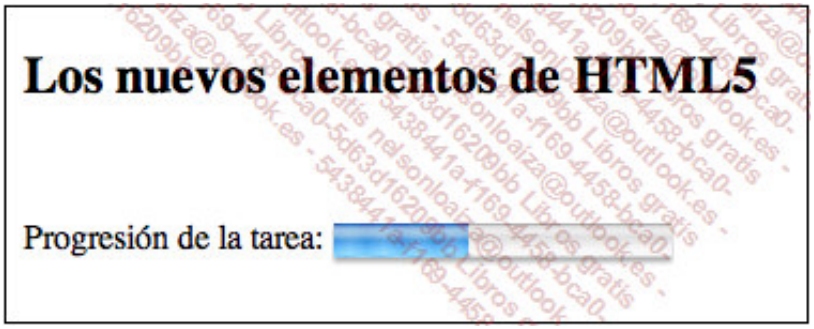
El nuevo elemento `<progress>` permite crear barras de progresión para las tareas. El movimiento de la barra de progresión se puede hacer con la ayuda de los atributos o bien con JavaScript.

Este sería un ejemplo de barra de progresión con dos atributos: `value`, para indicar el valor actual, y `max`, para indicar el valor máximo de la tarea.

```
<h2>Los nuevos elementos de HTML5</h2>
<p>Progresión de la tarea: <progress id="p" value="40" max="100">40%
```

```
</progress></p>
```

Nuevamente, el navegador podrá decidir cómo desea mostrarlo. Esta sería la visualización con Google Chrome:



9. Los elementos relacionados con los idiomas asiáticos

El elemento `<ruby>` le permite insertar texto en un idioma asiático. El elemento `<rt>` permite especificar la pronunciación de las palabras y el elemento `<rp>` permite insertar paréntesis a ambos lados de un texto en un idioma asiático para ocultar determinados caracteres.

10. Los elementos <figure> y <figcaption>

El elemento `<figure>` permite agrupar todos los elementos constitutivos para insertar una imagen: la imagen (elemento `<img>`) en sí misma (o una foto, vídeo, animación, etc.) y su leyenda con el nuevo elemento `<figcaption>`.

Este sería un ejemplo:

```
<p>Fotografía de una Sarracenia de mi jardín.<p/>
<figure>
    <img src="sarracenia.jpg" />
    <figcaption>sarracenia purpurea</figcaption>
</figure>
```

Así es como se vería:

Tipos de contenido de los elementos

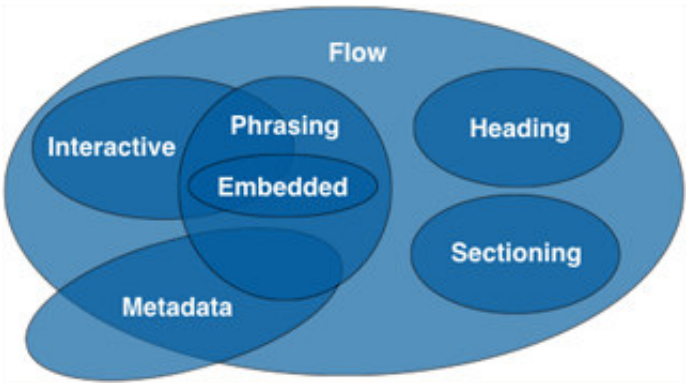
El HTML 4 establecía dos tipos principales de contenido: en línea (`inline`) y en bloque (`block`).

El HTML5 presenta una lista de tipos de contenido más completa (http://www.w3.org/TR/html5/dom.html#content-models). Además, determinados elementos pueden pertenecer a varios tipos diferentes, dependiendo del contexto en que se usen.

Tenemos:

- **Metadata content**: para los elementos de enlace entre los documentos, los elementos de presentación o los elementos de comportamiento del contenido (`base`, `command`, `link`,`meta`, `noscript`, `script`, `style` y `title`).
- **Flow content**: para los elementos utilizados en el cuerpo de la página (`a`, `article`,`blockquote`, `details`, `label`, `table`...).
- **Sectioning content**: para los elementos de la estructura (`article`, `aside`, `nav` y `section`).
- **Heading content**: para los elementos de encabezado de la sección (`h1` a `h6` y `hgroup`).
- **Phrasing content**: para los elementos de texto (`a`, `abbr`, `button`, `canvas`, `em`, `span`,`strong`...).
- **Embedded content**: para los elementos insertados (`audio`, `canvas`, `embed`, `iframe`, `img`,`math`, `object`, `svg` y `video`).
- **Interactive content**: para los elementos que impliquen interactividad con los usuarios (`a`,`audio`, `button`, `input`, `label video`...).

Así lo representa gráficamente el W3C:



Tenga en cuenta que el único objetivo de esta clasificación es el de organizar y clasificar los distintos elementos HTML. Esto no tendrá ningún efecto, ni determinará de ninguna manera la forma en la que se visualizarán dichos elementos (consulte el apartado siguiente). Lo que queremos decir es que esta clasificación no tendrá repercusiones reales en el diseño web de nuestras páginas web.

La visualización de los elementos

1. En HTML 4

El HTML 4 "ordenaba" los elementos en función del tipo de visualización en los navegadores (http://www.w3.org/TR/html4/struct/global.html#h-7.5.3). Los elemento de tipo `block` se visualizan a continuación unos de otros. Es el caso de los párrafos `<p>`, los títulos `<hx>`, las cajas `<div>`...

Los elementos de tipo `inline` se muestran uno al lado del otro, en la línea de texto. Es el caso de los vínculos `<a>`, de las divisiones `<span>`, de los estilos `<strong>`, `<em>`...

También tenemos la visualización: `inline-block`, `list-item`, `table`, `table-row`...

Las reglas de imbricación establecen que:

- un elemento `inline` solo puede contener otros elemento `inline` y datos, es decir, texto.
- un elemento `block` puede contener otros elemento `block`, así como elementos `inline`.

Sin embargo, con la propiedad `display` podemos cambiar sin problemas el tipo de visualización. Así, con `display: block` es posible visualizar un vínculo `<a>` como si fuera un bloque y de este modo presentará todas las características propias de los bloques.

2. En HTML5

En HTML5 esta "clasificación" ha quedado obsoleta. Es más, ya no se considera como un tipo de clasificación. Esto quiere decir que es usted, el autor del sitio web, quien deberá indicar cómo se deben visualizar los distintos elementos. De lo contrario, se aplicará la hoja de estilo predeterminada de cada navegador.

Usted podrá insertar sin problemas un elemento `<a>` que contenga un título `<h2>` y una imagen`<img>`, siempre y cuando especifique el modo de visualización de cada uno de los elementos en las hojas de estilo CSS con la propiedad `display`.

Esta estructura sería correcta en HTML5:

```
<a id="inicio" href="#">
    <h2 id="titulo">Inicio</h2>
    <img id="triángulo" src="triángulo.png" alt="" />
</a>
```

Los estilos CSS:

```
#inicio {
   display: block;
}
#titulo, #triangulo {
   display: inline;
}
```

Vista:

Los elementos de la estructura en HTML 4

1. Las cajas

En HTML 4, el elemento principal para estructurar las páginas web es la famosa "caja ". Con el elemento `<div>` es posible crear zonas de visualización de forma rectangular. Cada una de esas zonas, de esas cajas `<div>`, puede ser identificada de manera exclusiva, mediante un código de identificación, y de ese modo podemos aplicarle un formato con CSS. Ese código de identificación único se establece con el atributo `id`.

Ejemplo:

```
<div id="bannerSuperior">
  ...
</div>
```

También podemos usar las clases, cuando este formato se repita en la página y no se trate, por lo tanto, de un formato único.

Ejemplo:

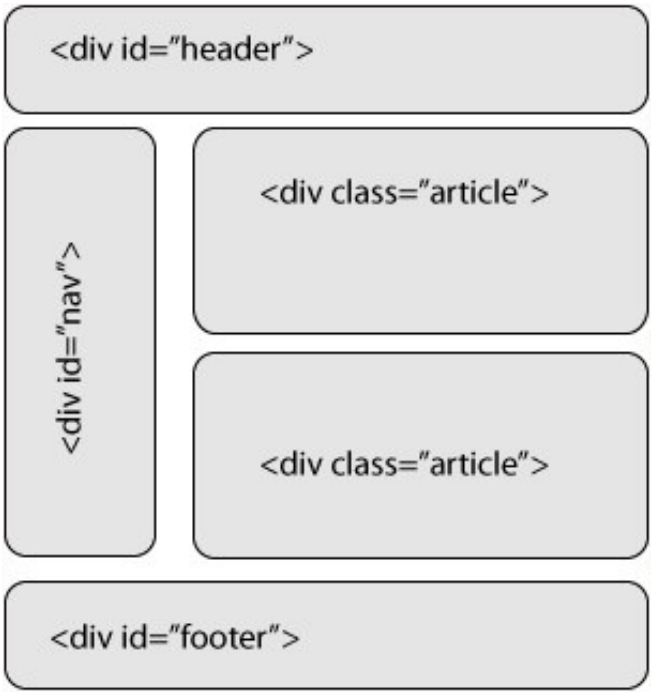
```
<div class="comentarios">
  ...
</div>
```

A continuación, una vez que haya determinado la estructura, podrá aplicar el diseño con las hojas de estilo CSS, mediante los selectores de identificación que correspondan al código de identificación o a la clase en cuestión.

Ejemplo:

```
#bannerSuperior {
  ...
}
.comentarios{
  ...
}
```

En el siguiente ejemplo podemos ver la estructura de una página de tipo blog con cajas `<div>`:



2. La "divitis"

Con HTML 4 y CSS 2.1, la estructura de las páginas, cuando esta es relativamente compleja, puede requerir una gran cantidad de cajas `<div>`. Podemos llegar a crear páginas que contengan decenas y decenas de `<div>`. Si usted cae en ese exceso, habrá contraído la enfermedad llamada "divitis", es decir, el uso excesivo de cajas `<div>`.

3. Un contenido no semántico

Todas esas cajas `<div>` presentan otro problema: el de la semántica de los contenedores. Cada caja `<div>` se distingue de las demás gracias a su código de identificación único, que el diseñador web le habrá asignado en función de su "humor" o de sus "ganas" en ese momento.

Por ese motivo, las cajas `<div>` no son semánticas: el contenido esperado no está definido por ningún parámetro. Una caja `<div>` identificada con `id="izquierda"` no nos aporta ningún dato sobre su contenido. Podría tratarse de una barra de navegación, de información legal o de cualquier otro tipo de contenido.

Ahora bien, la evolución del HTML se dirige hacia un mayor uso de la semántica.

Los elementos de la estructura en HTML5

1. Los nuevos elementos de la estructura

Con el HTML5 llegan nuevos elementos de estructura semántica. Estos nuevos elementos han sido definidos y se les ha asignado un nombre tras un largo análisis de los nombres más usados en las cajas `<div>`.

2. El elemento <header>

El nuevo elemento `<header>` permite insertar una zona de visualización para las cabeceras. Puede definirse esta cabecera para toda la página o para una zona determinada: artículo, menú lateral...

Debemos considerar que este elemento se puede usar desde dos puntos de vista:

- a nivel de la página: es la típica cabecera de la página, que a menudo aparece en lo alto de la pantalla, con un logotipo, un eslogan, un menú de navegación principal...
- a nivel del contenido: permite disponer de una zona de introducción del contenido que se incluya a continuación.

Esta es la definición del W3C para el elemento `<header>`: "*The header element represents a group of introductory or navigational aids. A header element is intended to usually contain the section's heading (an h1-h6 element or an hgroup element), but this is not required. The header element can also be used to wrap a section's table of contents, a search form, or any relevant logos.*"

Como puede ver, el elemento `<header>` no tiene que aparecer obligatoriamente en la estructura de la página.

3. El elemento <footer>

El nuevo elemento `<footer>` permite insertar una zona de visualización para los pie de página. Volvemos a encontrarnos con la misma semántica de los encabezados. Es posible definir un pie de página para la página completa, o bien para cualquier otra zona de visualización o para un artículo.

Esta es la definición del W3C: "*The footer element represents a footer for its nearest ancestor sectioning content or sectioning root element. A footer typically contains information about its section such as who wrote it, links to related documents, copyright data, and the like*."

Su uso es similar al de los `<header>`, pero aplicado a los pie de página. Por lo tanto, no debemos seguir al "pie de la letra" la traducción literal de "pie de página". Se trata más bien de un "pie de zona de visualización", donde la zona de visualización puede ser una página, una sección, un artículo...

4. El elemento <nav>

El elemento `<nav>`, como su nombre indica, está pensado para la visualización de una zona de navegación con vínculos hipertexto. Cuidado, no quiere decir que solo podrá tener una zona de navegación en cada página, o que tenga que crear tantos elementos `<nav>` como zonas de navegación haya en sus páginas, bastará con que los identifique correctamente. El elemento `<nav>`está pensado en especial para la navegación principal del sitio web, para la inserción de vínculos entre las páginas de dicho sitio web.

La definición del W3C es bastante clara: "*The nav element represents a section of a page that links to other pages or to parts within the page: a section with navigation links. Note: Not all groups of links on a page need to be in a nav element - the element is primarily intended for sections that consist of major*

navigation blocks. In particular, it is common for footers to have a short list of links to various pages of a site, such as the terms of service, the home page, and a copyright page. The footer element alone is sufficient for such cases; while a nav element can be used in such cases, it is usually unnecessary."

5. El elemento <section>

El elemento `<section>` permite agrupar elementos que tengan la misma temática. De este modo podemos agrupar en un mismo elemento un contenido, con su título y su pie de página.

Esta es la definición del W3C: "*The section element represents a generic section of a document or application. A section, in this context, is a thematic grouping of content, typically with a heading*."

6. El elemento <article>

El elemento `<article>` permite insertar un contenido autónomo, que puede volver a usarse en otro lugar del sitio web, sin que por ello se anule su significado.

Esta es la definición del W3C: "*The article element represents a self-contained composition in a document, page, application, or site and that is, in principle, independently distributable or reusable, e.g. in syndication. This could be a forum post, a magazine or newspaper article, a blog entry, a user-submitted comment, an interactive widget or gadget, or any other independent item of content*."

Claro está, un artículo podría tener una cabecera (`<header>`), un pie de página (`<footer>`) y varios títulos (`<hx>`).

7. El elemento <aside>

El elemento `<aside>` permite mostrar un contenido relacionado con el contenido al cual esté vinculado. Puede tratarse de barras laterales (*sidebars*), de zonas de widgets, de complementos sobre artículos, etc.

8. El elemento <main>

Último en llegar a los borradores del W3C, el elemento `<main>` permite representar el contenido principal de la página.

La definición del W3C lo indica bien: *"The main element represents the main content of the body of a document or application. The main content area consists of content that is directly related to or expands upon the central topic of a document or central functionality of an application"*.

De esta forma, una página podrá estar perfectamente estructurada con un elemento `<main>` para el contenido de la página, un encabezamiento `<header>`, contenidos con elementos `<article>` y un pie de página `<footer>`.

El W3C indica una serie de restricciones para el uso del elemento `<main>`:

- un solo elemento `<main>` por página,
- no debe utilizarse dentro (elemento descendente) de los elementos `<article>`, `<aside>`, `<footer>`, `<header>` o `<nav>`,
- debería estar asociado con el `role main` (`<main role="main">`) para una mejor accesibilidad de los sitios Web.

9. Las cajas

Y, por último, quiero aclarar que aunque use HTML5 no tiene por qué erradicar el uso de las cajas`<div>`. ¡Todavía podemos usarlas y siguen teniendo su utilidad!

El atributo semántico "role"

El XHTML (http://www.w3.org/1999/xhtml/vocab/#XHTMLRoleVocabulary) y el XHTML2 *Working Group*preconizaban el uso del atributo `role` (http://www.w3.org/TR/xhtml2/mod-roleAttribute.html) para definir de forma más semántica los elementos estructurales de una página web.

En HTML5 podemos usar el atributo `role` para incluir esa información adicional gracias al módulo **WAI-ARIA** (http://www.w3.org/TR/aria-in-html/ en working Draft a 3 de octubre de 2013). Este módulo se ocupa de la gestión del contenido de las páginas web para la personas discapacitadas. Determinados elementos HTML5 tienen un `role` implícito, como, por ejemplo, el elemento `<nav>`, cuyo `role`implícito es `navigation`.

Estos son los principales valores del atributo `role`:

- `main`: define el contenido principal de un documento.
- `secondary`: define una parte secundaria del documento.
- `navigation`: define la barra de navegación del documento.
- `banner`: aparece por lo general en lo alto de la página y suele contener el logotipo y el eslogan de la empresa.
- `contentinfo`: indica que dicho elemento aporta información sobre el contenido de la página (autores, copyrights, menciones legales...).
- `definition`: presenta la definición de un elemento.
- `note`: corresponde, por lo general, a una nota entre paréntesis o al final de la página.
- `seealso`: indica que el elemento contiene información relacionada con el contenido principal de la página.
- `search`: contiene el formulario de búsqueda de una página web.

Veamos un ejemplo simple de cómo usarlo:

```
<div id="buscar" role="search">
 ...
</div>
```

Otro ejemplo:

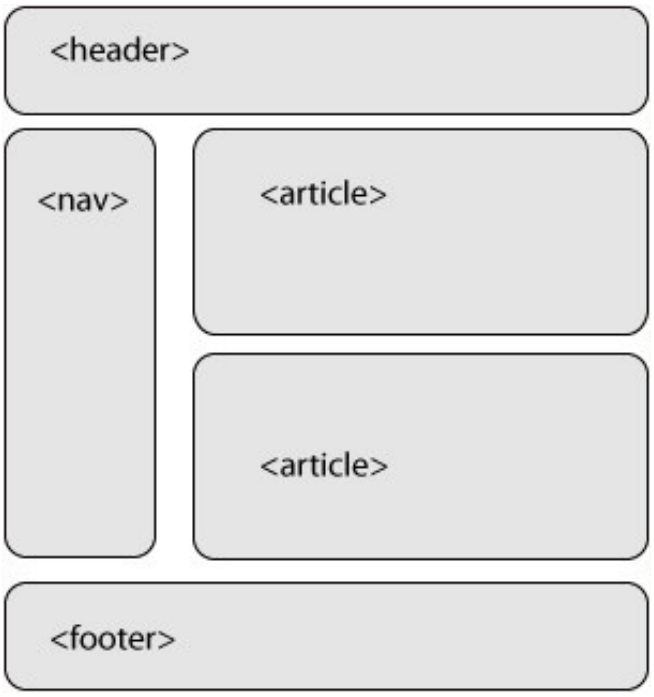
```
<header id="banner" role="banner">
 ...
</header>
```

Si desea obtener más información sobre **Accessible Rich Internet Applications (WAI-ARIA)**, consulte la **Candidate Recommendation** del 18 de enero de 2011: http://www.w3.org/TR/2011/CR-wai-aria-20110118/

Ejemplos de estructura en HTML5

1. Una estructura simple

Si volvemos a tomar como ejemplo la estructura de página que usamos al principio de este capítulo, podremos modificarla para adaptarla al HTML5.



En el elemento `<header>` encontraremos los elementos del encabezado de la página, como el logotipo y el eslogan, por ejemplo.

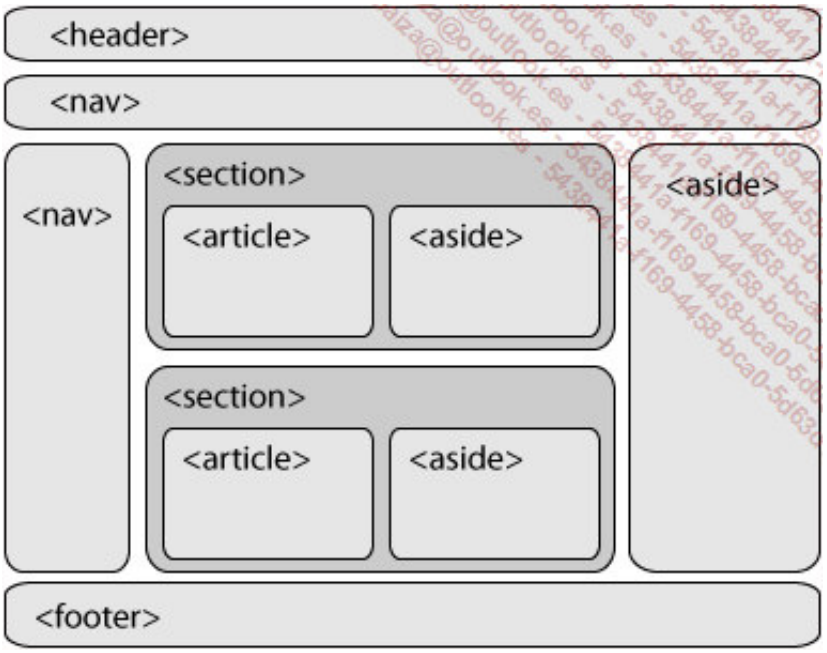
En el elemento `<nav>`, situado a la izquierda, encontraremos los vínculos que nos permitirán navegar por ese sitio web.

Todos los artículos del blog estarán colocados, por supuesto, dentro de los elementos `<article>`.

Por último, el pie de página, `<footer>`, podrá contener las menciones legales o los vínculos de contacto, por ejemplo.

2. Una estructura más elaborada

Veamos ahora la estructura de un sitio web un poco más elaborado.



Encontramos de nuevo el elemento `<header>`, que ya todos conocemos.

Debajo tenemos el primer elemento `<nav>`, para el menú de navegación general del sitio web, que nos permitirá navegar por las distintas páginas.

A la izquierda tenemos una segunda caja `<nav>`, para la navegación secundaria, que contiene los vínculos relacionados directamente con el contenido de la página en cuestión.

A la derecha encontramos el elemento `<aside>`, que muestra información relacionada con el contenido de la página, como los vínculos promocionales o los contenidos relacionados, por ejemplo.

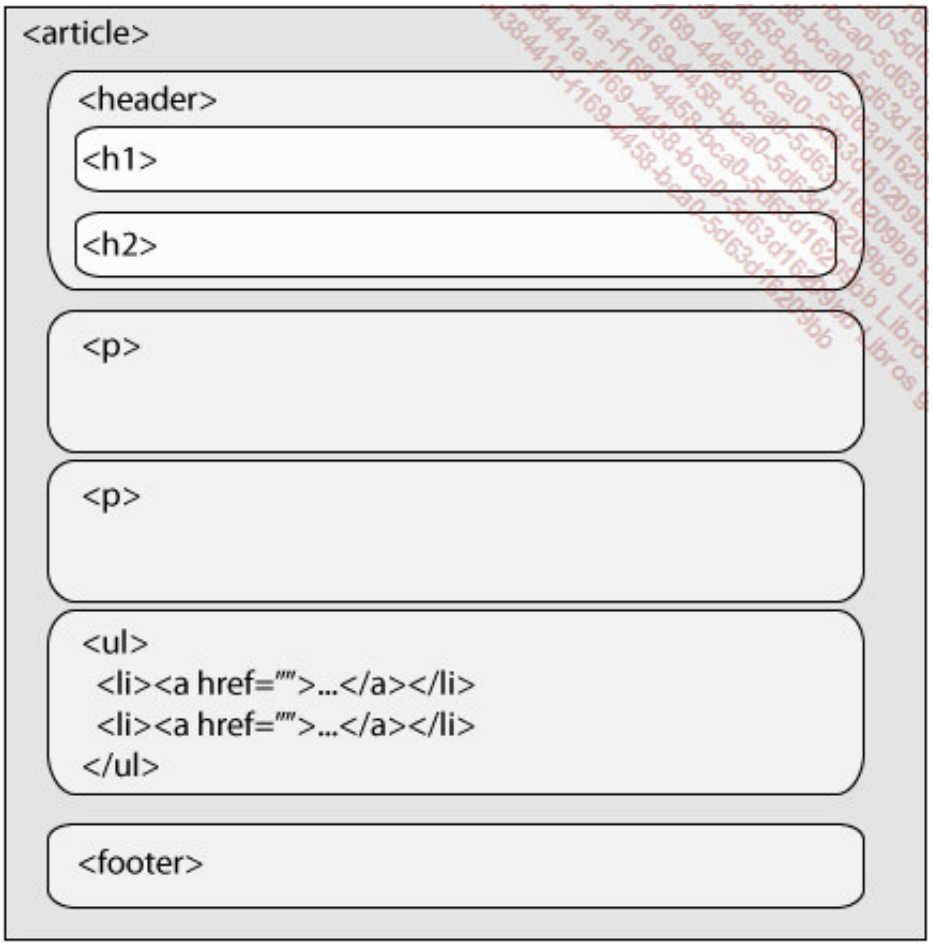
El contenido de la página aparece dentro de dos elementos `<section>`, de modo que podamos distinguir con facilidad esos dos contenidos diferentes. Cada `<section>` contiene un elemento`<article>` para el contenido textual y un elemento `<aside>` para incluir los elementos de información adicional relacionados con el artículo (iconografía, enlaces...).

Por último, la página presenta un pie de página `<footer>` que contiene la información legal, las condiciones de venta, un vínculo de contacto, un plano de acceso...

Claro está, cada elemento de la estructura deberá contar con un código de identificación único o con una clase común para varios elementos.

3. La estructura de un artículo

Veamos ahora la estructura de lo que podría ser un artículo de un sitio web en HTML5.



Tenemos un elemento `<article>` como contenedor general.

Nuestros artículos presentan cabeceras, introducciones, por lo que hemos usado el elemento`<header>`. Este elemento `<header>` contiene el título `<h1>` del artículo y su subtítulo `<h2>`.

El contenido textual del artículo se sitúa entre los elementos `<p>`. El artículo incluye vínculos, que completan el contenido facilitado, ordenados en una lista `<ul>`.

Por último, el artículo presenta un pie de página `<footer>` o, mejor dicho, un "pie de artículo", con la fecha de la publicación, la sección a la que pertenece dicho artículo y el nombre del autor, por ejemplo.

Las técnicas de formato

Las técnicas para dar formato a la página, con las cajas flotantes o las formas de posicionamiento, se pueden aplicar perfectamente a los nuevos elementos HTML5.

Plantillas para sitios web en HTML5

Desde que se publicó HTML5, multitud de diseñadores web han creado plantillas de diseño de sitios web (en inglés, *templates*) que utilizan HTML5.

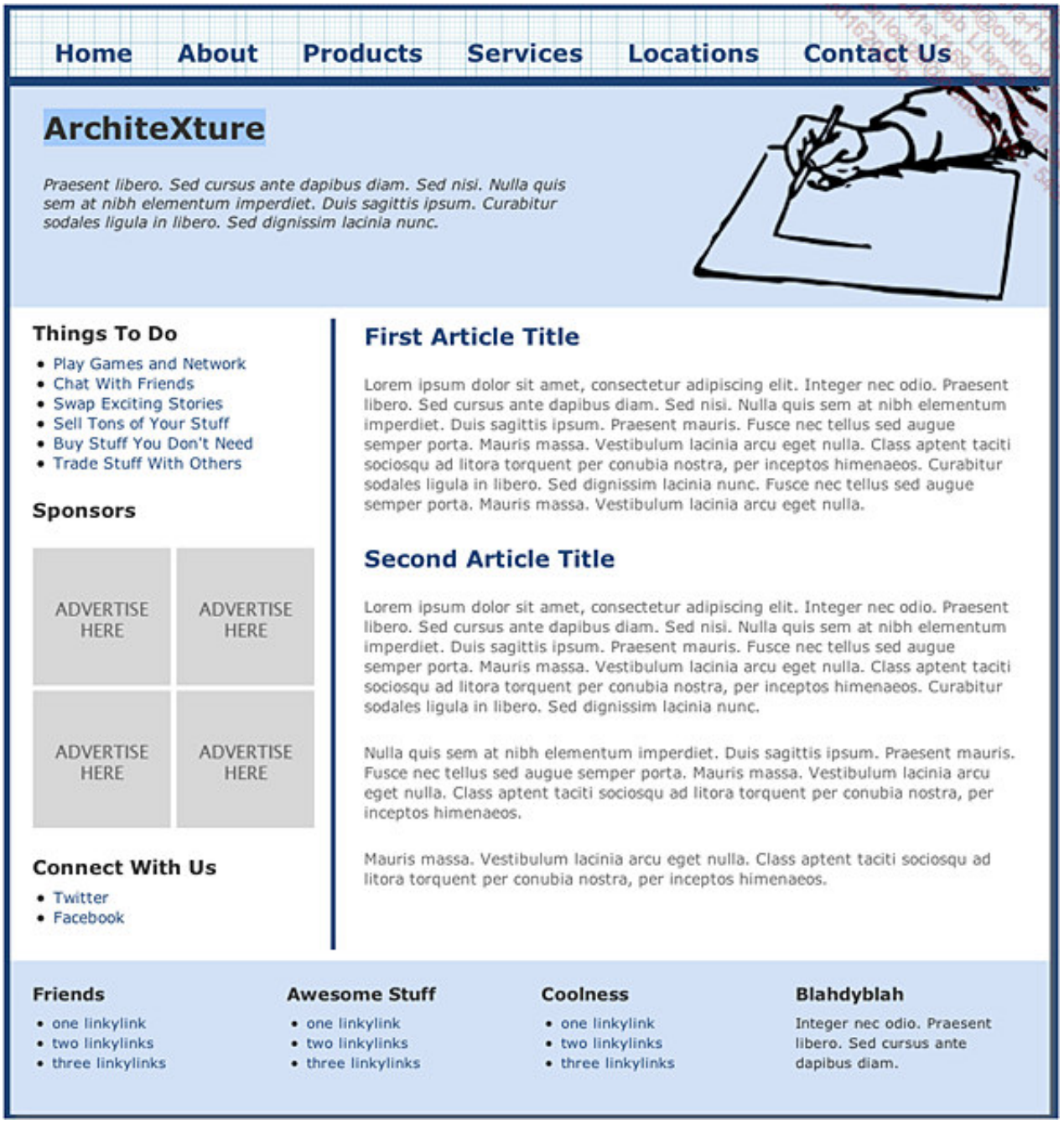
La plantilla ArchiteXture

1. La fuente

Esta es la URL de la plantilla: http://freehtml5templates.com/architexture-html5-and-css3-template/.
A partir de esta dirección podremos probarla en línea y descargar los archivos fuente.

2. El diseño del sitio web

Observe el diseño de esta plantilla, que como ve, es bastante sobrio.



Podemos distinguir fácilmente las cuatro "zonas" de visualización del sitio web:

- el menú de navegación en la parte superior,
- el banner de presentación del sitio web, que contiene el nombre, un eslogan y un logotipo,

- la zona central, que muestra información general en el lado izquierdo y los artículos en la parte derecha,
- el pie de página, que contiene los vínculos, en la parte inferior.

3. La estructura general

La estructura general del sitio web refleja fielmente la presentación visual.

Tenemos una caja `<div id="wrapper">` que sirve de contenedor general y que permite centrar el sitio web en la ventana del navegador.

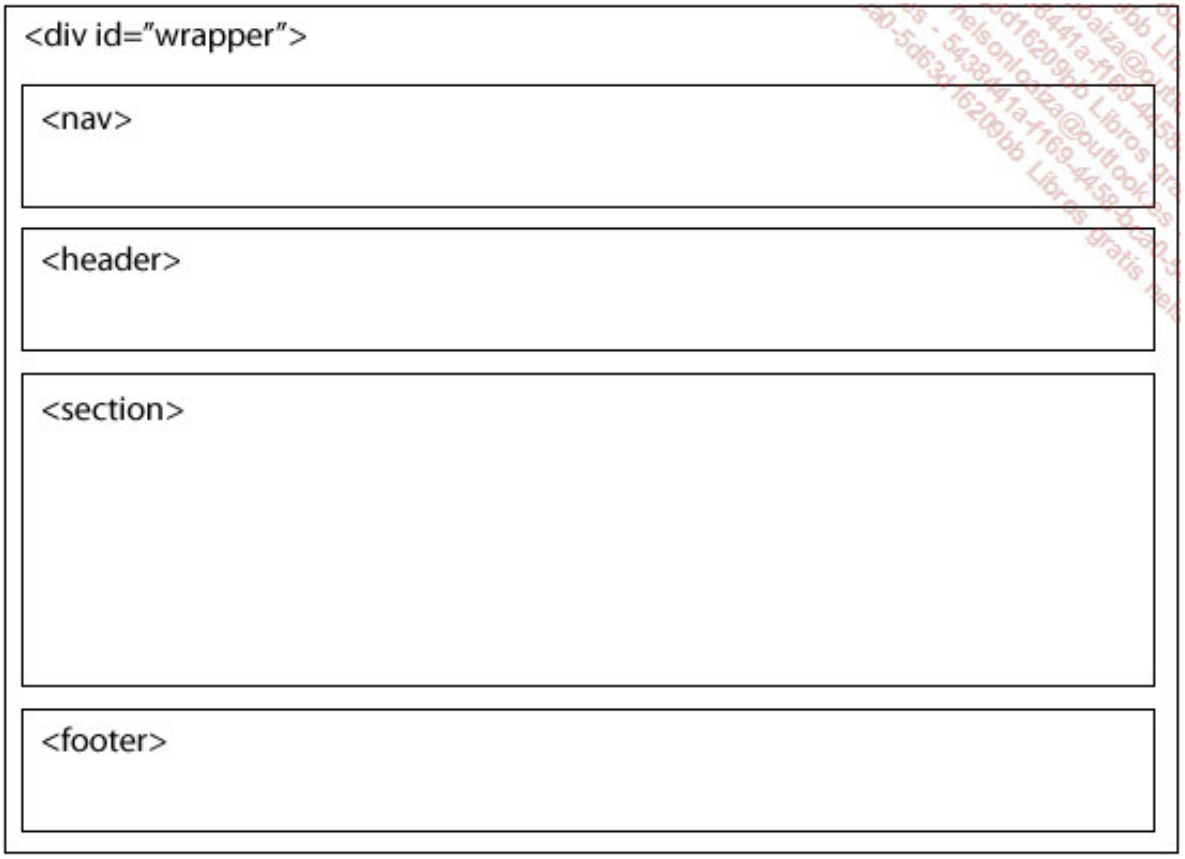
Encontramos a continuación el elemento `<nav>`, que contiene el menú de navegación de la parte de arriba. El diseñador ha empleado correctamente el elemento `<nav>`, ya que lo ha usado para insertar el menú de navegación principal del sitio web.

La información general del sitio web se ha insertado en un elemento `<header>`. Resulta bastante acertado, ya que se trata de información general que se mostrará en todas las páginas del sitio web. Efectivamente, se trata de una cabecera de página de sitio web.

La zona central del sitio web es un elemento `<section>`. Una sección es una zona de visualización que reúne elementos con una temática similar. En el ejemplo se trata, simplemente, de mostrar esos elementos en la zona central de la página.

El pie de página es, lógicamente, un elemento `<footer>`.

Veamos la estructura de las cajas:



4. El código de la estructura

Veamos el código HTML5 de la estructura de este sitio web:

```
<!doctype html>
<html lang="en">
<head>
   <meta charset="utf-8" />
   <title>ArchiteXture</title>
   <link rel="stylesheet" href="styles.css" type="text/css"
media="screen" />
   <link rel="stylesheet" type="text/css" href="print.css"
media="print" />
   <!--[if IE]><script
src="http://html5shiv.googlecode.com/svn/trunk/html5.js">
</script><![endif]-->
</head>
<body>
<div id="wrapper">
   <nav>
      ...
   </nav>
   <header>
      ...
   </header>
   <section id="main">
      ...
   </section>
   <footer>
      ...
   </footer>
</div>
</body>
</html>
```

5. El menú de navegación

El menú de navegación (<nav>) contiene una caja que incluye la clásica lista (<ul>) de vínculos.

Este es el código:

```
<nav><!-- top nav -->
    <div class="menu">
       <ul>
            <li><a href="#">Home</a></li>
            <li><a href="#">About</a></li>
            <li><a href="#">Products</a></li>
            <li><a href="#">Services</a></li>
            <li><a href="#">Locations</a></li>
            <li><a href="#">Contact Us</a></li>
       </ul>
    </div>
 </nav><!-- end of top nav -->
```

La visualización del menú de navegación:

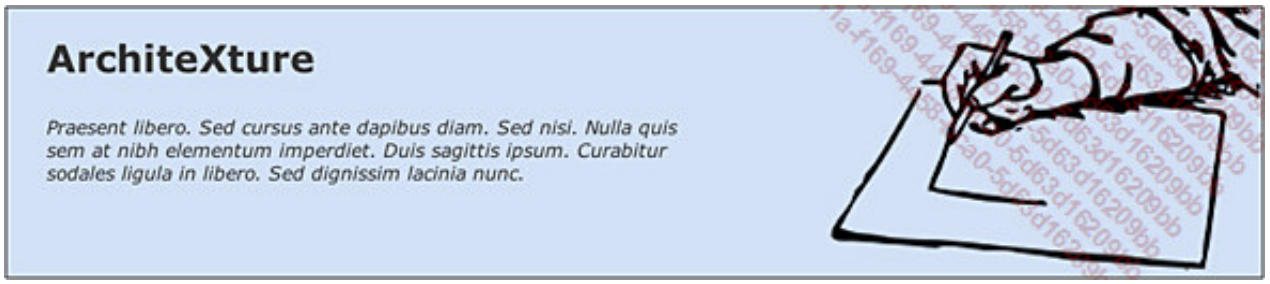
6. El banner de presentación

El banner de presentación (<header>) contiene una caja con el logotipo, un título de nivel 1 (<h1>) y un párrafo.

Este es el código:

```
<header><!-- header -->
    <div id="plandesign"><img src="images/plans.png" alt=""/></div>
    <h1><a href="#">ArchiteXture</a></h1>
    <p>Praesent libero...</p>
 </header><!-- end of header -->
```

La visualización del banner de presentación:



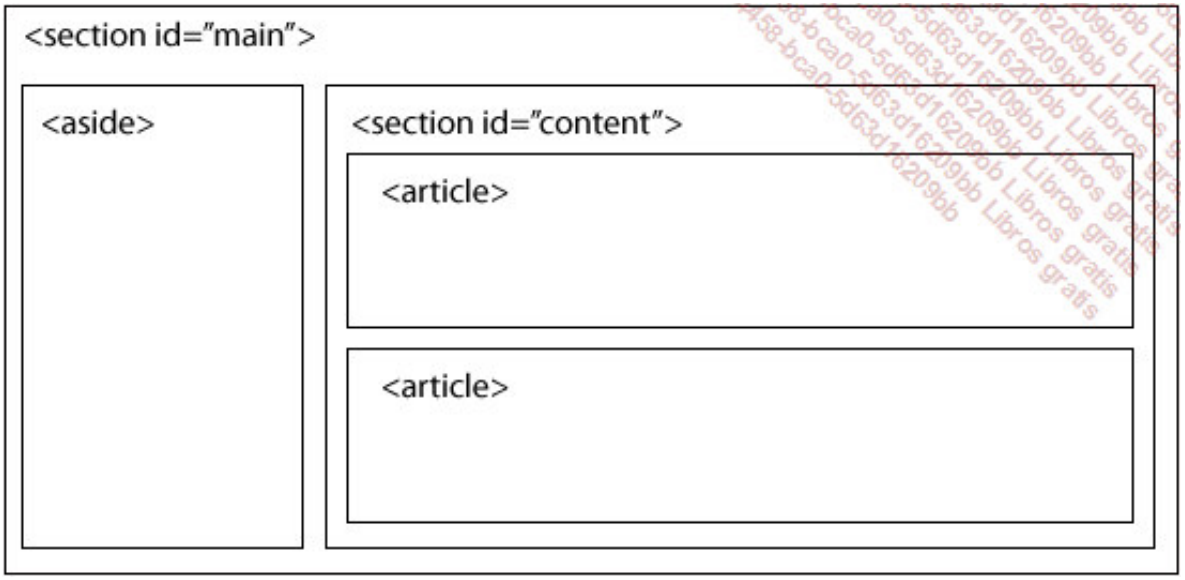
7. La zona central

La zona central del sitio web se ha insertado en un elemento <section id="main">. Este contiene a su vez otra caja <section id="content"> para mostrar el contenido propiamente dicho del sitio web, los artículos de la parte derecha, y un elemento <aside id="sidebar"> para mostrar información general en la parte izquierda.

El elemento <section id="content"> contiene, lógicamente, los elementos <article> para los distintos artículos del sitio web. Cada artículo contiene títulos <h2> y párrafos <p>.

El elemento <aside> contiene diversos elementos: títulos <h3>, listas <ul> e imágenes <img>.

Esta es la estructura de las cajas:



Veamos el código de la zona central:

```
<section id="main"><!-- #main content and sidebar area -->
    <section id="content"><!-- #content -->
        <article>
```

```
            <h2><a href="#">First Article Title</a></h2>
            <p>Lorem ipsum dolor...</p>
        </article>
        <article>
            <h2><a href="#">Second Article Title</a></h2>
            <p>Lorem ipsum...</p>
        </article>
    </section><!-- end of #content -->
    <aside id="sidebar"><!-- sidebar -->
        <h3>Things To Do</h3>
        <ul>
            <li><a href="#">Play Games and Network</a></li>
            <li><a href="#">Chat With Friends</a></li>
        </ul>
        <h3>Sponsors</h3>
        <img src="images/ad125.jpg" alt="" />
        <img src="images/ad125.jpg" alt="" /><br />
        <h3>Connect With Us</h3>
        <ul>
            <li><a href="#">Twitter</a></li>
            <li><a href="#">Facebook</a></li>
        </ul>
    </aside><!-- end of sidebar -->
</section>
```

Así se presenta la zona central, `<section id="main">`:

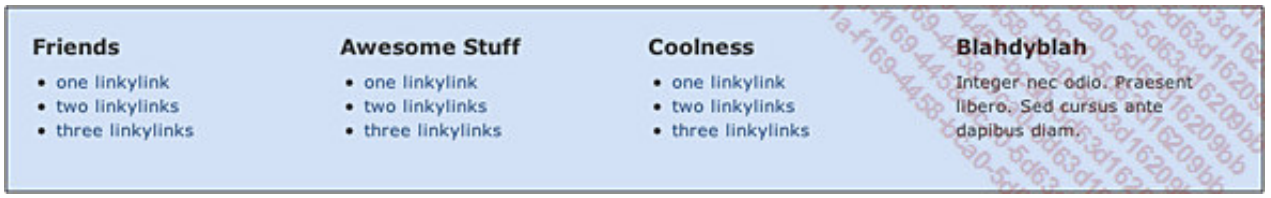
8. El pie de página

El pie de página utiliza el elemento `<footer>`. Este contiene dos cajas `<div>` para obtener el diseño deseado. Dentro de la segunda caja `<div>` encontramos cuatro elementos `<aside>`, que corresponden a las cuatro zonas de visualización.

Este es el código:

```
<footer>
    <section id="footer-area">
        <section id="footer-outer-block">
            <aside class="footer-segment">
                <h4>Friends</h4>
                <ul>
                     <li><a href="#">one linkylink</a></li>
                     <li><a href="#">two linkylinks</a></li>
              </ul>
            </aside><!-- end of #first footer segment -->
            <aside class="footer-segment">
                  ...
            </aside><!-- end of #second footer segment -->
            <aside class="footer-segment">
                  ...
            </aside><!-- end of #third footer segment -->
            <aside class="footer-segment">
                  ...
            </aside><!-- end of #fourth footer segment -->
        </section><!-- end of footer-outer-block -->
    </section><!-- end of footer-area -->
</footer>
```

Y el diseño obtenido:



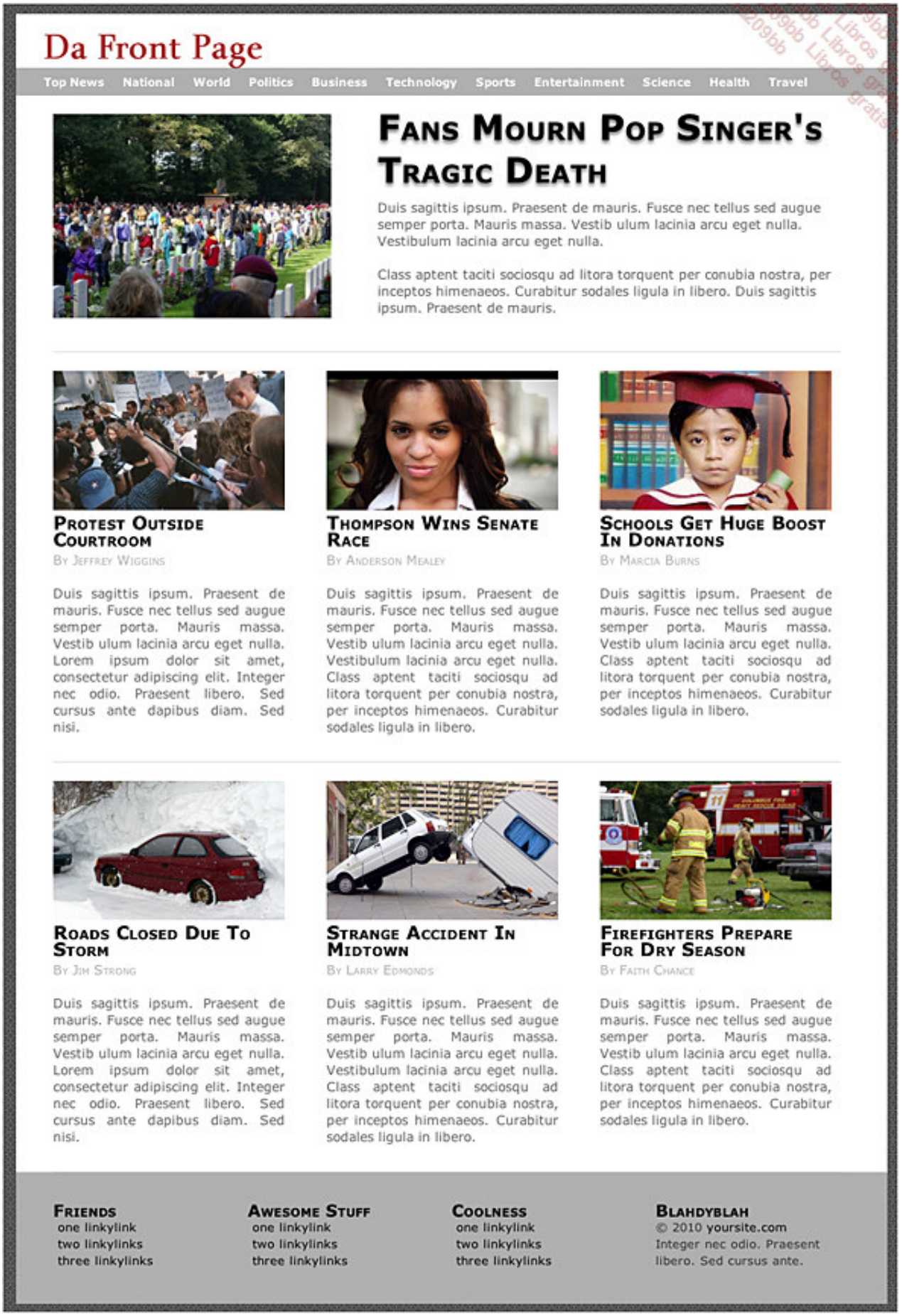
La plantillaDaFrontPage

1. La fuente

Esta es la URL de dicha plantilla: http://freehtml5templates.com/dafrontpage-html5-and-css3-template/. A partir de esta dirección podremos probarla en línea y descargar los archivos fuente.

2. El diseño del sitio web

Así se presenta esta plantilla, al estilo de una revista.



3. La estructura general

La estructura general del sitio web aparece dentro de una caja `<div id="wrapper">`.

El título del sitio web se encuentra dentro de un elemento de título de nivel 1: <h1>.

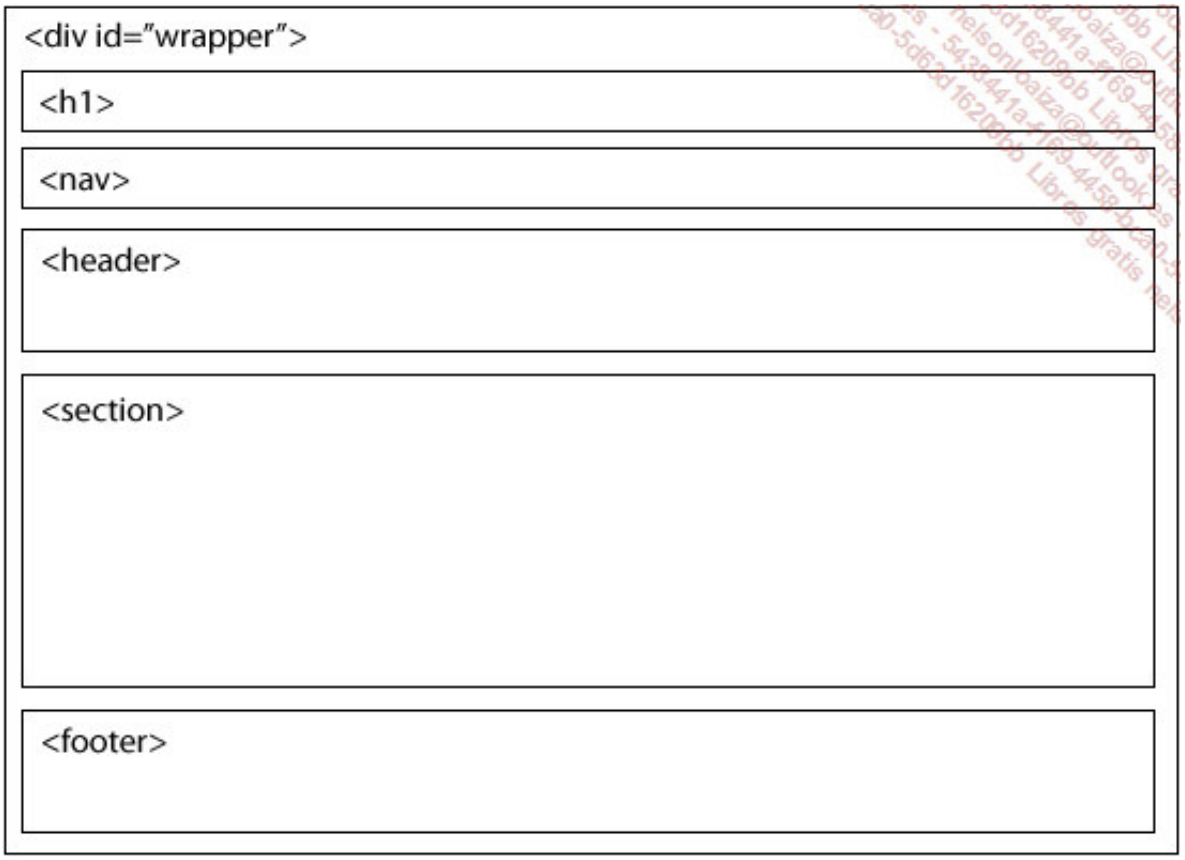
El menú de navegación se incluye, como cabía esperar, dentro de un elemento <nav>.

El artículo resaltado en la parte superior se encuentra dentro del elemento <header> de la página.

La estructura de tres columnas se encuentra dentro de un elemento <section>, ya que las tres columnas presentan un contenido similar.

Por último, el pie de página aparece, claro está, dentro de un elemento <footer>.

Veamos la estructura de las cajas:



Veamos el código de la estructura general:

```
<!doctype html>
<html lang="en">
<head>
     <meta charset="utf-8" />
     <title>Da Front Page</title>
     <link rel="stylesheet" href="styles.css" type="text/css"
media="screen" />
     <link rel="stylesheet" type="text/css" href="print.css"
media="print" />
     <!--[if IE]><script
src="http://html5shiv.googlecode.com/svn/trunk/html5.js">
</script><![endif]-->
</head>
<body>
<div id="wrapper">
     <h1><a href="#">Da Front Page</a></h1>
     <nav>
        ...
     </nav>
     <header>
        ...
```

```
    </header>
    <section id="main">
       ...
    </section>
    <footer>
      ...
    </footer>
</div>
</body>
</html>
```

4. El título del sitio web

El título del sitio web ha sido insertado dentro de un elemento <h1>.

```
<h1><a href="#">Da Front Page</a></h1>
```

5. El menú de navegación

El menú de navegación aparece lógicamente dentro de un elemento <nav>. En una caja , todos los vínculos están ordenados con la clásica lista <ul>.

Este es el código que se ha usado:

```
<nav>
      <div class="menu">
          <ul>
              <li><a href="#">Top News</a></li>
              <li><a href="#">National</a></li>
              ...
          </ul>
      </div>
</nav>
```

La visualización del menú de navegación:

6. El encabezado del sitio web

En el elemento <header> encontramos el artículo "en portada". Es el diseñador del sitio web quien ha querido mostrar un artículo como cabecera del sitio web. Siguiendo la lógica semántica, este artículo se mostrará en todas las páginas del sitio web.

El elemento <header> contiene dos cajas : una para mostrar la imagen y otra para el contenido textual.

Este es el código que se ha usado:

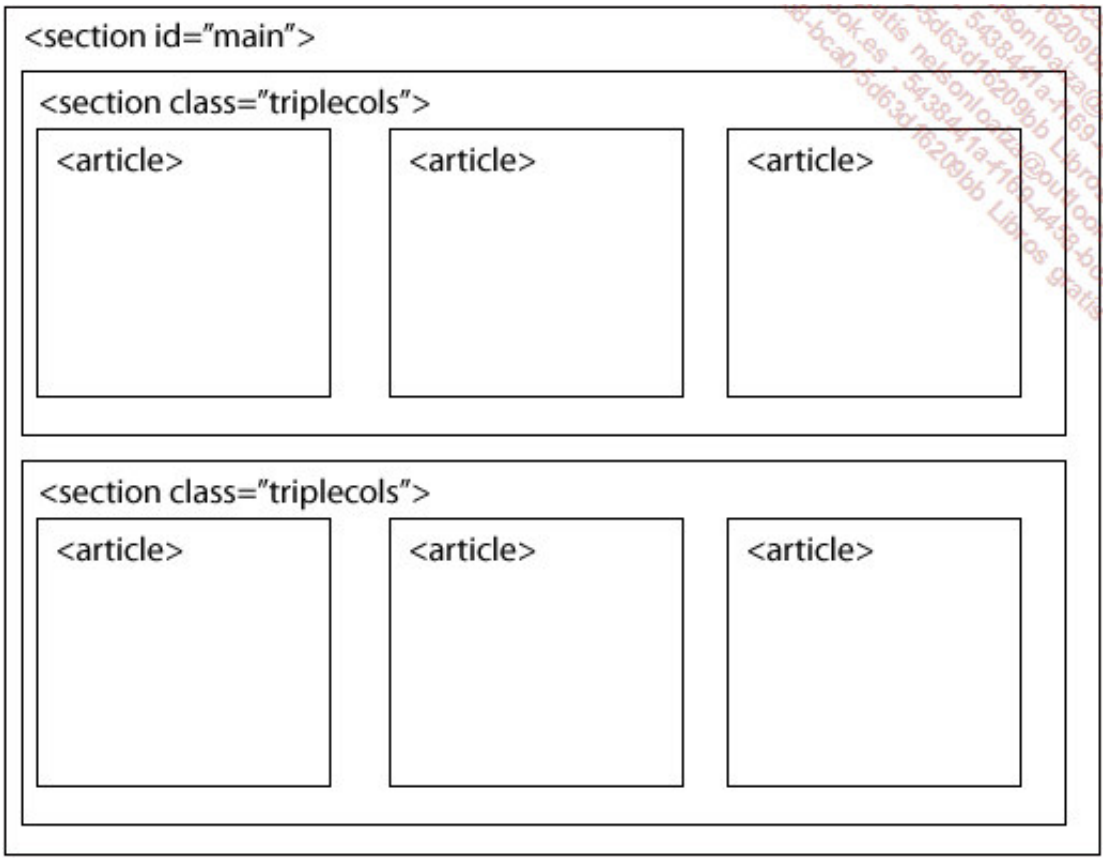
```
<header>
    <div class="headlineimage">
       <img src="images/headlineimg.jpg" alt="" />
    </div>
    <div class="headline">
       <h2><a href="#">Fans Mourn Pop Singer's Tragic Death</a></h2>
       <p>Duis sagittis ipsum...</p>
       <p>Class aptent...</p>
    </div>
</header>
```

7. La zona central

La zona central del sitio web, que contiene los artículos en columnas, utiliza el elemento<section id="main">. Este contiene dos elementos <section class="triplecols">para presentar las dos series de artículos en tres columnas.

Cada elemento <section class="tripelcols"> contiene tres elementos <article>. Y cada elemento <article> contiene, como es lógico, un artículo, que incluye elementos <a> para los vínculos, <img> para las imágenes y <p> para los párrafos.

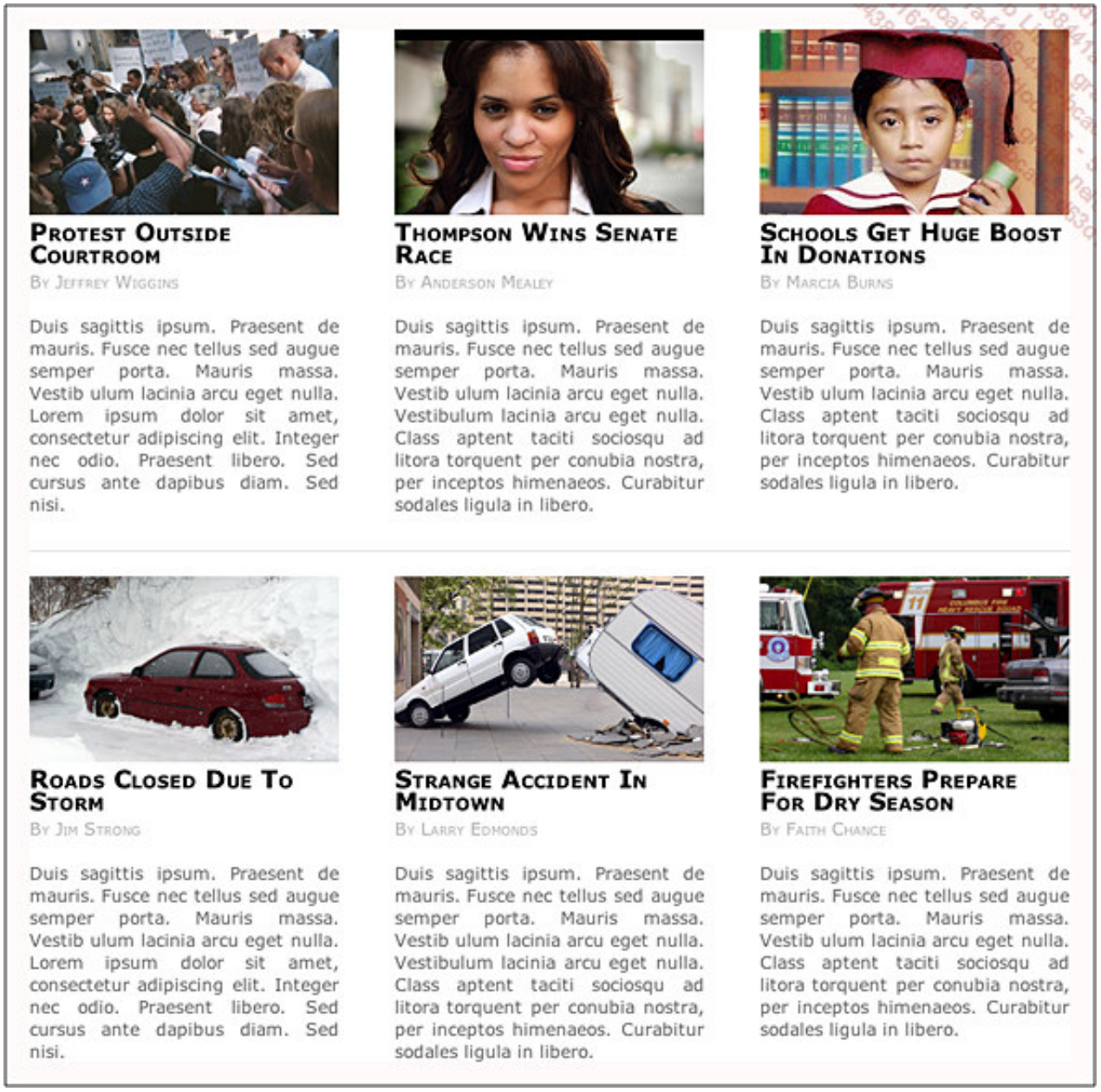
Veamos la estructura de los artículos en columnas:



Este es el código que se ha usado:

```
<section id="main">
    <section class="triplecols">
        <article class="tripleblocks tripleleftblock">
           <a href="#">
               <img class="thumbnail".../>
               <span class="caption"><b>Protest Outside
                     Courtroom</b>
               </span>
           </a>
           <p class="byline">By Jeffrey Wiggins</p>
           <p>Duis sagittis ipsum...</p>
        </article>
        <article>
        ...
        </article>
        <article>
        ...
        </article>
    </section>
    <section class="triplecols">
        <article>
        ...
        </article>
        <article>
        ...
        </article>
        <article>
        ...
        </article>
    </section>
</section>
```

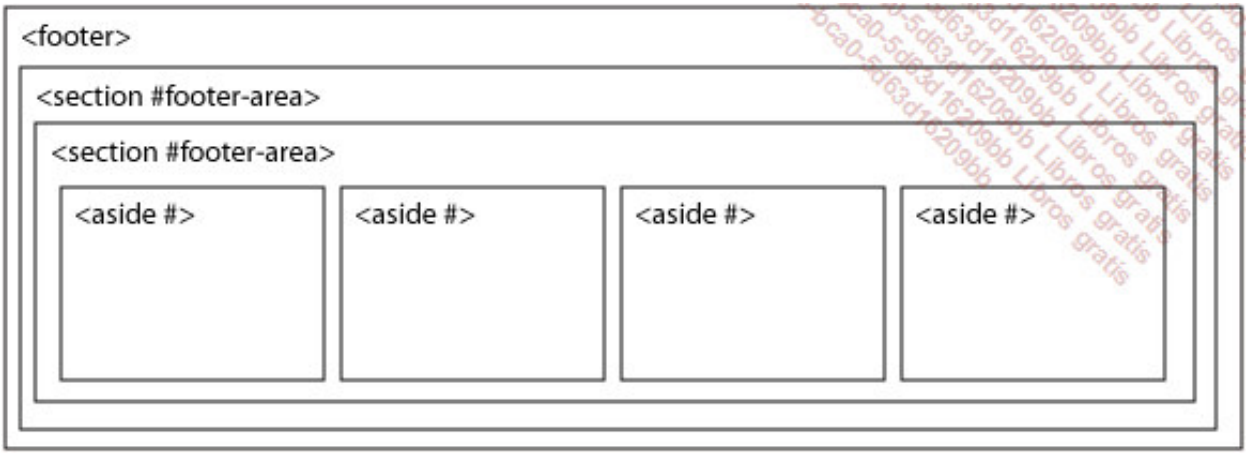
Así se presenta la zona central con los artículos:



8. El pie de página

El pie de página `<footer>` usa dos cajas `<div>` para su presentación. Cada una de las cuatro zonas de visualización utiliza un elemento `<aside>`.

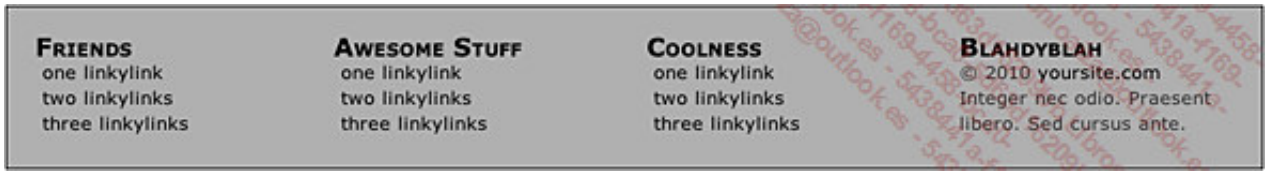
Esta es la estructura del pie de página:



Y este es el código que se ha usado:

```
<footer>
    <section id="footer-area">
        <section id="footer-outer-block">
            <aside class="footer-segment first">
                <h3>Friends</h3>
                <ul>
                    <li><a href="#">one linkylink</a></li>
                    <li><a href="#">two linkylinks</a></li>
                    ...
                </ul>
            </aside>
            <aside class="footer-segment second">
               ...
            </aside>
            <aside class="footer-segment third">
               ...
            </aside>
            <aside class="footer-segment last">
               ...
            </aside>
        </section>
    </section>
</footer>
```

Así se visualiza el pie de página:



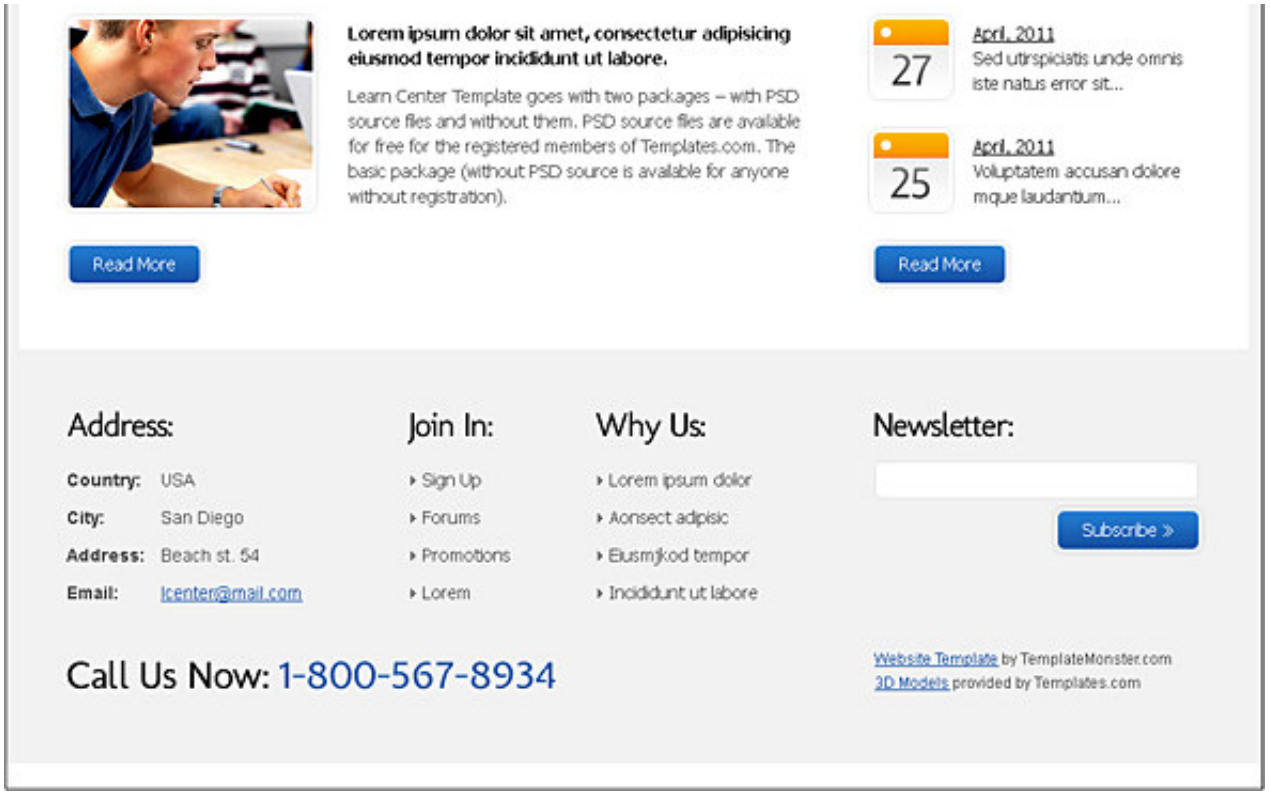
La plantilla Learning Center

1. La fuente

Esta es la URL de dicha plantilla: http://www.templatemonster.com/free-templates/free-website-template-learning-center.php. A partir de esta dirección podremos probarla en línea y descargar los archivos fuente.

2. El diseño del sitio web

Veamos el diseño de esta plantilla, de tipo portal web para centros educativos.

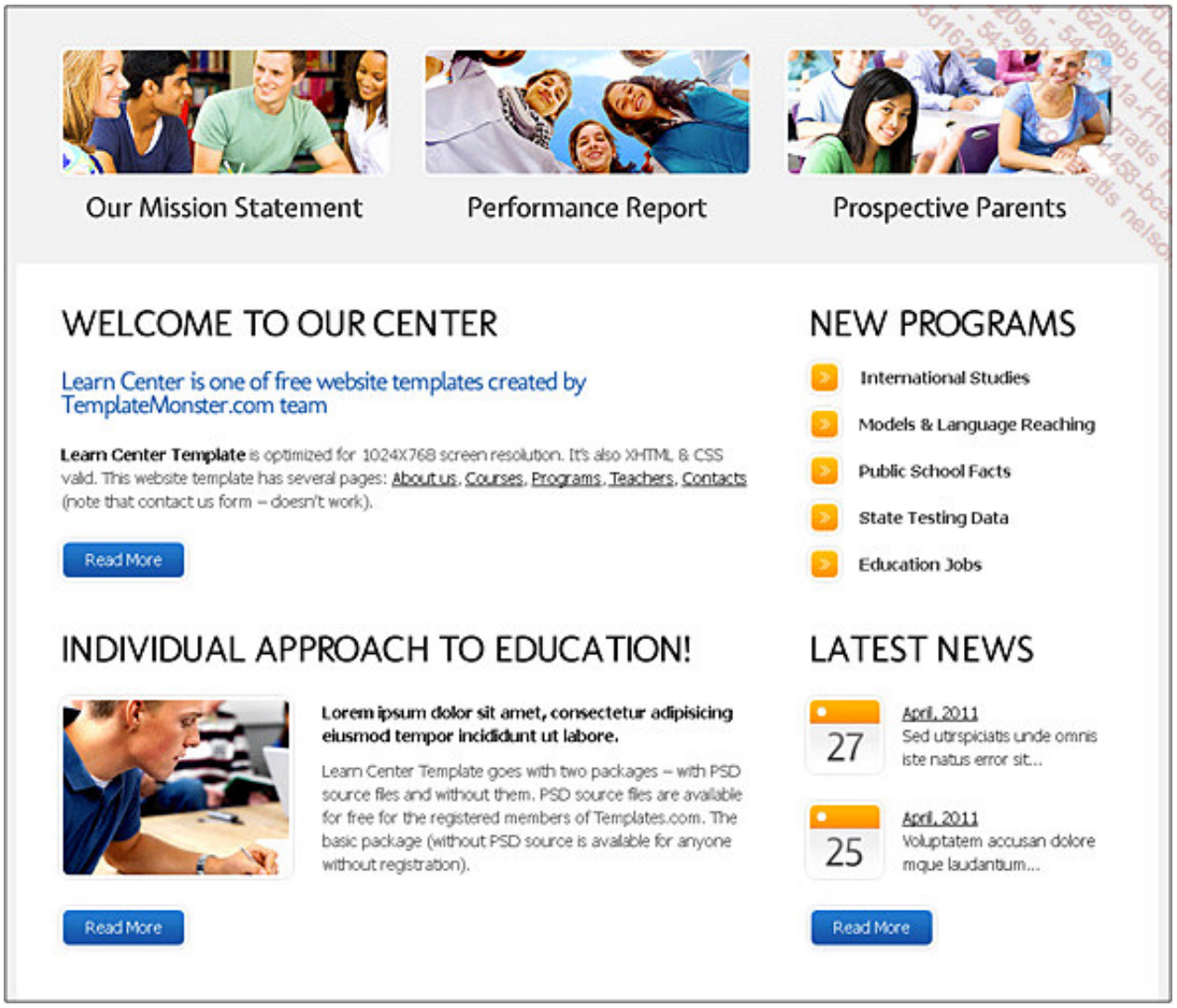


3. La estructura general

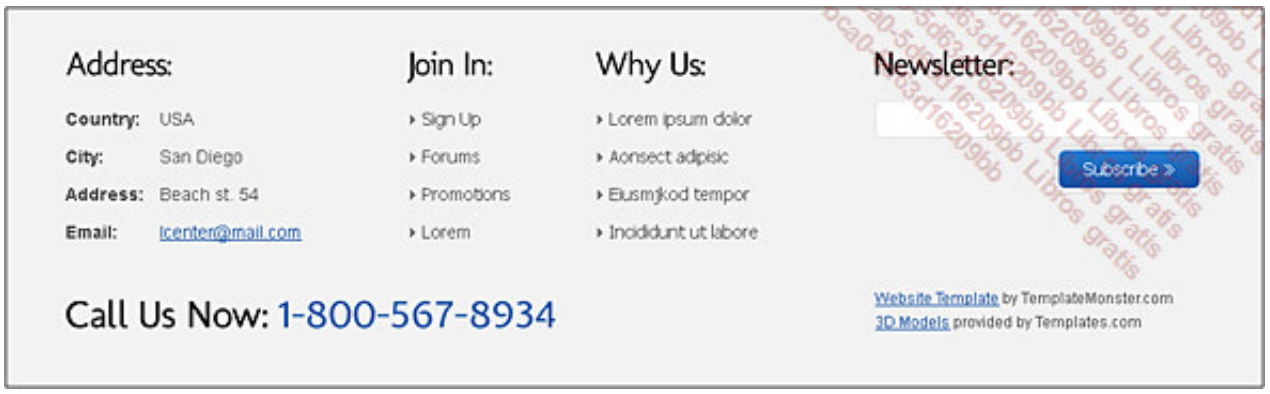
El sitio web se compone de dos cajas `<div>`, que utilizan dos clases CSS.

La primera caja `<div class="body1">` permite visualizar la parte superior del sitio web. Esta contiene a su vez una caja `<div class="main">` que se ha usado para aplicar el diseño. En esta caja encontramos un elemento `<header>` que corresponde a la parte superior del sitio web, la cabecera.

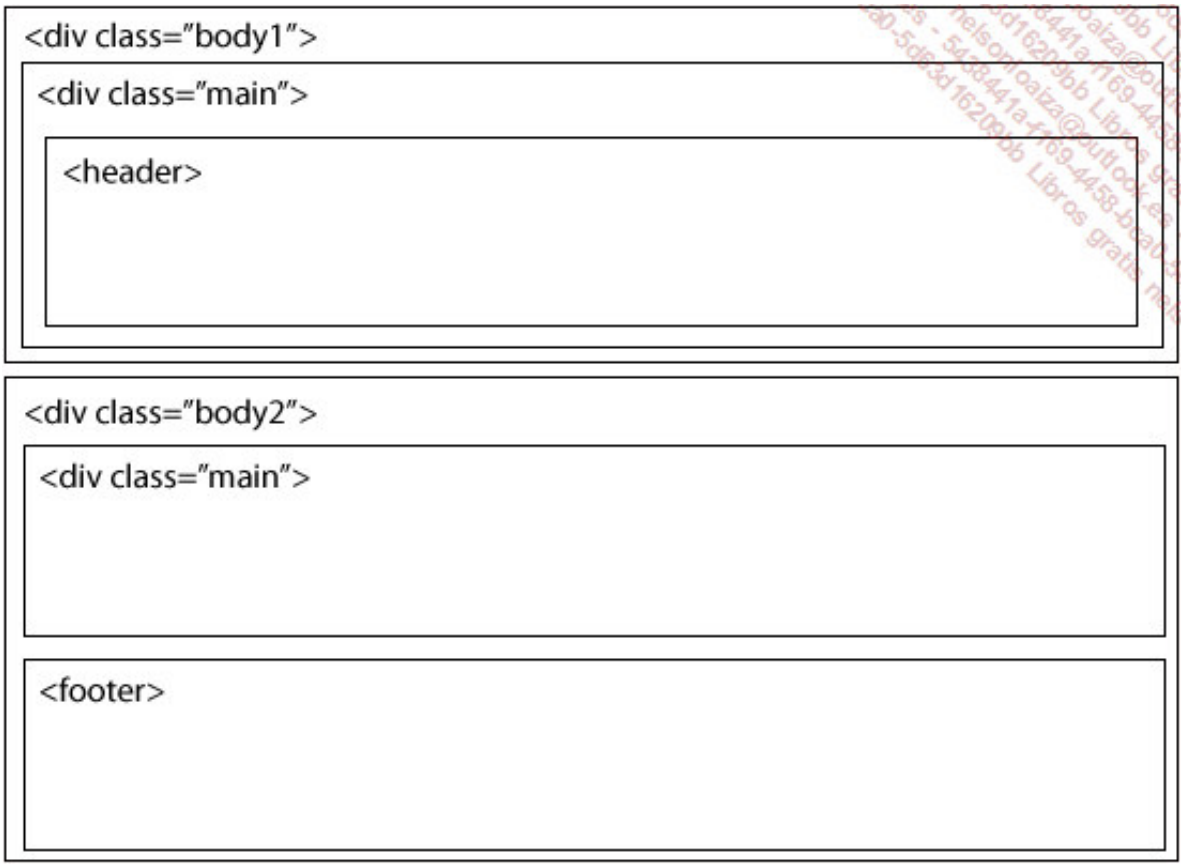
La segunda caja `<div class="body2">` permite visualizar la parte inferior del sitio web. Esta contiene a su vez una caja `<div class="main">` que se ha usado para aplicar el diseño. Esta caja contiene un elemento `<section id="content">` para la visualización de la zona central del sitio web.



A continuación tenemos una caja `<footer>` para la visualización del pie de página del sitio web.



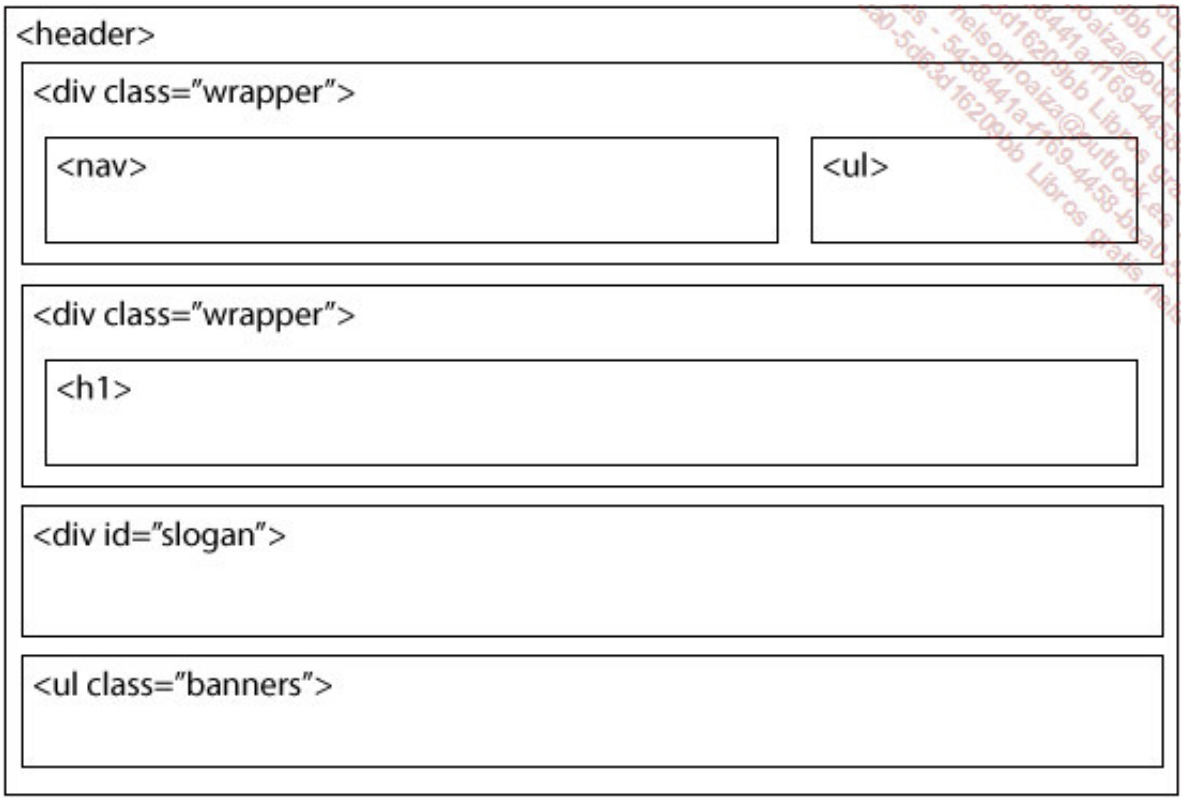
Esta es la estructura general:



4. La cabecera

Como acabamos de ver, el elemento `<header>` se encuentra dentro de dos cajas `<div>`. Este elemento `<header>` contiene a su vez tres cajas `<div>` para las tres partes de la cabecera y una lista `<ul>`:

- el menú de navegación,
- el título "Learn Center",
- el eslogan,
- los tres grandes botones en una columna.



5. El menú de navegación

El menú de navegación utiliza, como cabía esperar, el elemento `<nav>`, que contiene una lista `<ul>`con todos los vínculos. Los tres vínculos de la derecha para las redes sociales se incluyen en otra lista `<ul>`.

Este es el código de la primera caja `<div class="wrapper">`:

```
<div class="wrapper">
   <nav>
      <ul id="menu">
         <li><a href="index.html">About</a></li>
         <li><a href="Courses.html">Courses</a></li>
         <li><a href="Programs.html">Programs</a></li>
         <li><a href="Teachers.html">Teachers</a></li>
         <li><a href="Admissions.html">Admissions</a></li>
         <li class="end"><a href="Contacts.html">Contacts</a></li>
      </ul>
   </nav>
   <ul id="icon">
      <li><a href="#"><img src="images/icon1.jpg" alt=""></a></li>
      <li><a href="#"><img src="images/icon2.jpg" alt=""></a></li>
      <li><a href="#"><img src="images/icon3.jpg" alt=""></a></li>
   </ul>
</div>
```

6. La continuación de la cabecera

A continuación, la cabecera presenta las típicas cajas `<div>`, un título `<h1>` y una lista `<ul>`:

```
<div class="wrapper">
   <h1><a href="index.html" id="logo">Learn Center</a></h1>
</div>
<div id="slogan">
   We Will Open The World<span>of knowledge for you!</span>
</div>
<ul class="banners">
   <li><a href="#"><img src="images/banner1.jpg" alt=""></a></li>
   <li><a href="#"><img src="images/banner2.jpg" alt=""></a></li>
   <li><a href="#"><img src="images/banner3.jpg" alt=""></a></li>
</ul>
```

7. Los artículos resaltados

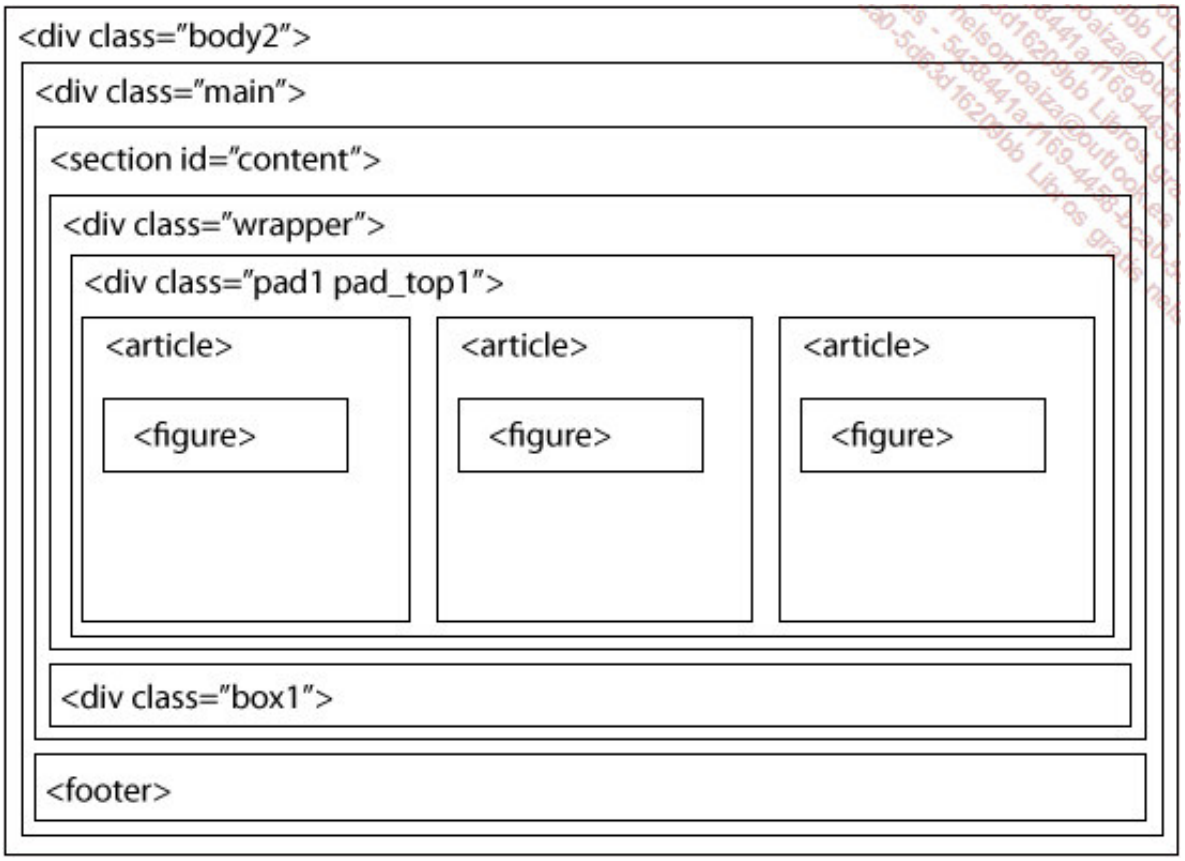
Veamos con mayor detalle la estructura que permite visualizar los artículos resaltados en la parte superior de la zona central.

La zona central del sitio web es un elemento `<section id="content">` que se encuentra dentro de dos cajas `<div>`.

Los tres artículos resaltados en la parte superior de la zona central se visualizan lógicamente gracias a tres elementos `<article>`.

La imagen que ilustra cada artículo se ha insertado gracias al elemento `<figure>`.

Como puede ver, esta plantilla utiliza correctamente los nuevos elementos de estructura del HTML5.



Este es el código que se ha usado para la presentación de los tres artículos resaltados:

```
<div class="body2">
    <div class="main">
       <section id="content">
           <div class="wrapper">
              <div class="pad1 pad_top1">
                 <article class="cols marg_right1">
                   <figure><a href="#"><img
src="images/page1_img1.jpg" alt=""></a></figure>
                  <span class="font1">Our Mission Statement</span>
                 </article>
                 <article class="cols marg_right1">
                    ...
                 </article>
                 <article class="cols">
                    ...
                 </article>
              </div>
           </div>
```

8. La continuación de la zona central

La continuación de la zona central sigue la misma lógica, con los elementos `<article>` y`<figure>`.

9. El pie de página

En el pie de página también se usan los elementos `<article>` para cada uno de los seis "bloques" de visualización. Personalmente, yo habría usado elementos `<section>`.

Address:

Country: USA

City: San Diego

Address: Beach st. 54

Email: Icenter@mail.com

Join In:

- Sign Up
- Forums
- Promotions
- Lorem

Why Us:

- Lorem ipsum dolor
- Aonsect adipisic
- Eiusmjkod tempor
- Incididunt ut labore

Newsletter:

Subscribe »

Call Us Now: 1-800-567-8934

WordPress y las plantillas HTML5

WordPress (http://es.wordpress.org/) es a día de hoy el CMS (*Content Management System* o sistema de gestión de contenido) más popular para la creación y la administración de sitios web. La presentación visual de los sitios WordPress se gestiona mediante plantillas de diseño que se conocen como **themes**. La versión actual de WordPress, la 3.6, propone un tema que utiliza HTML5 de forma predeterminada, el tema **twentythirteen**.

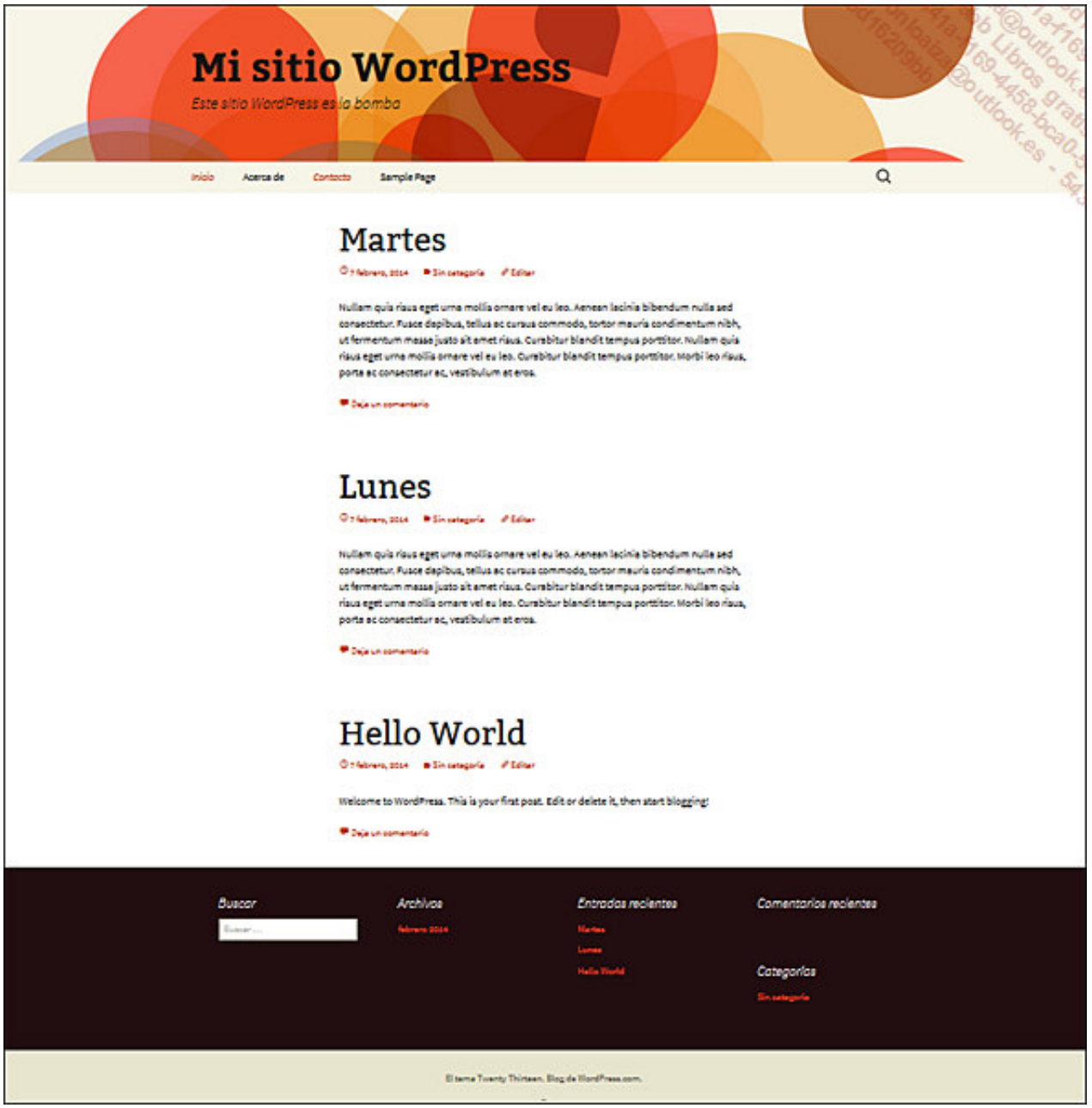
No vamos a hablar aquí del funcionamiento de las páginas web dinámicas en PHP de Wordpress, sino que nos limitaremos al resultado HTML5 obtenido.

Si desea aprender a usar WordPress, le recomiendo que consulte los libros, publicados también con la editorial ENI:

- WordPress 3.5 - Un CMS para crear y administrar blogs y sitios web
- WordPress 3.5 - Las mejores extensiones

1. El tema twentythirteen

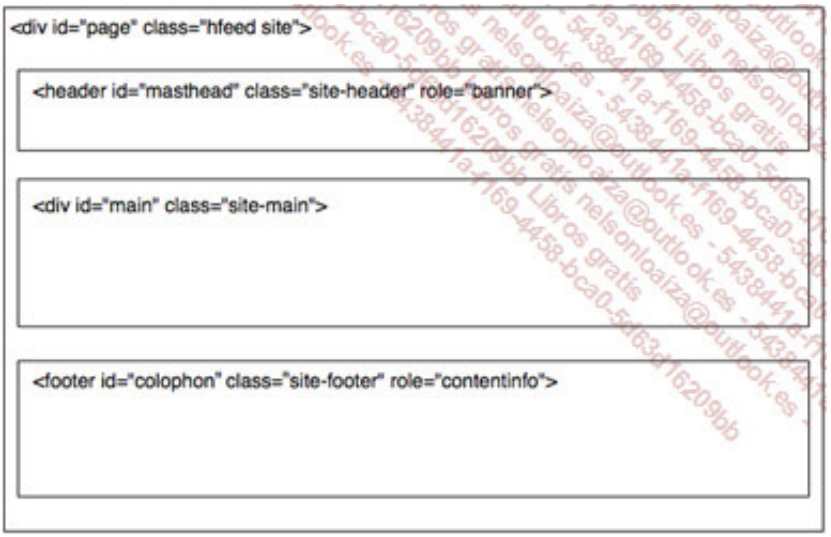
El tema predeterminado de WordPress 3.6 es **twentythirteen** y está estructurado en HTML5.



La estructura general del tema es la siguiente:

Tenemos una caja `<div id="page">` que agrupa toda la estructura. Dentro de ella, tenemos tres cajas:

- una caja `<header id="masthead">` que contiene el encabezado del sitio con el vínculo a la página de inicio y la barra de navegación. Tenemos un `role` definido.
- una caja `<div id="main">` con el contenido del sitio.
- una caja `<footer id="colophon">` con el pie de página y los widgets. Tenemos también un `role` definido.



2. El encabezado

El encabezado del sitio WordPress contiene una imagen de fondo y el título con un vínculo para volver a la página de inicio del sitio. Debajo se encuentra la barra de navegación y el cuadro de búsqueda de texto.

La caja `<div>` general del identificador `masthead` con un `role` definido en `banner`. Contiene dos elementos: un vínculo a la página de inicio (`<a class="home-link">`) y una caja `<div>` para navegar y buscar (`<div id="navbar">`).

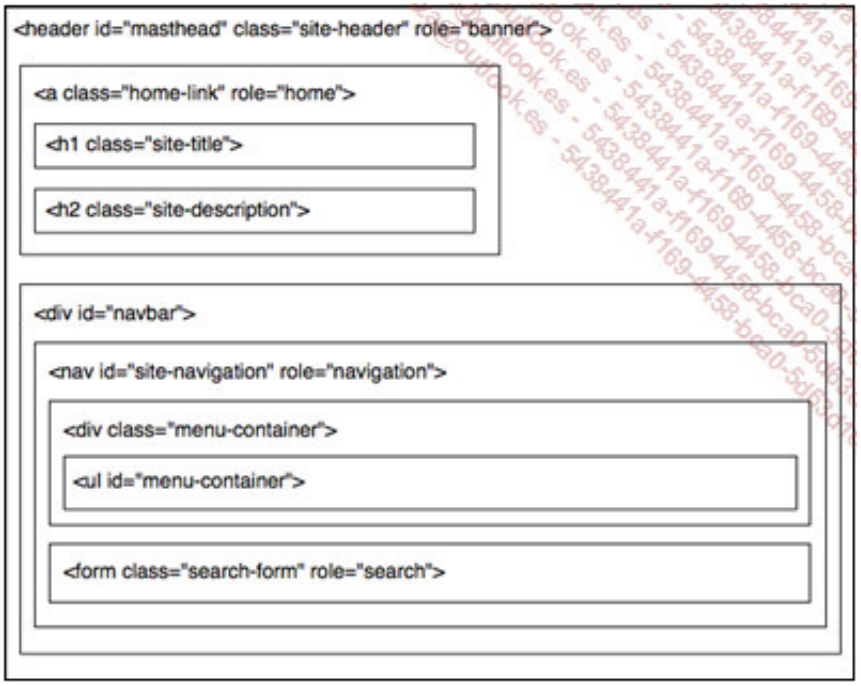
El elemento `<a>` que permite regresar a la página de inicio contiene dos elementos de título: un`<h1>` para el nombre del sitio y un `<h2>` para su descripción. No olvide que en HTML5 podemos imbricar sin problemas elementos de visualización habitualmente en bloques en elementos de visualización habitualmente en línea, si el diseñador define bien los valores de la propiedad CSS`display`. En este tema, el elemento `<a>` tiene `display: block` como propiedad CSS.

La caja `<div id="navbar">` es la que contiene la barra de navegación.

La barra de navegación está situada en el elemento `<nav id="site-navigation">` (con un`role` definido en `navigation`) y el campo de búsqueda en `<form class="search-form"`

`role="search">`.

Esta es la estructura del encabezamiento:



3. La lista de entradas

La página de inicio «clásica» de los sitios web WordPress muestra una lista con las últimas entradas publicadas.

Esta es la estructura de la página de inicio:

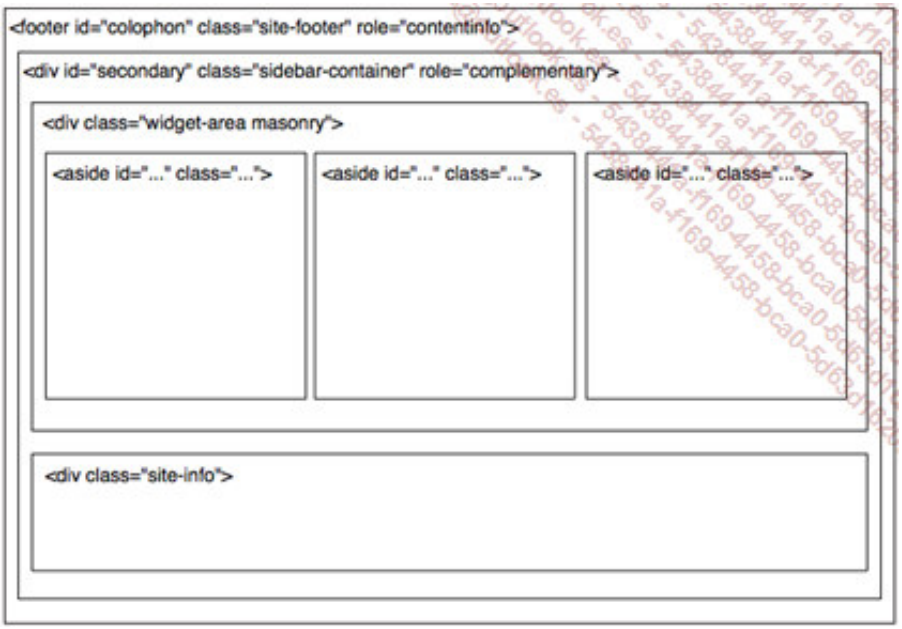
La caja principal es `<div id="main" class="site-main">`, contiene dos cajas más `<div>:id="primary"` y `id="content"`. Esta última alberga la lista de entradas de la página de inicio.

Cada entrada del sitio se encuentra en un elemento `<article>`. Cada entrada posee un identificador único y numerosas clases para darle formato. Cada entrada contiene un título y metadatos, que son mostrados en un elemento `<header>`. El contenido de las entradas, su redactado, se sitúa en la caja `<div class="entry-content">`. Por último, el link a los comentarios se sitúa en el elemento `<footer class="entry-meta">`.

4. El pie de página

El pie de página del tema **twentythirteen** contiene la zona de los widgets de WordPress y la información sobre el uso de WordPress.

Esta es su estructura:



El elemento principal es el `<footer>` de la página: `<footer id="colophon" class="site-footer" role="contentinfo">`. Este elemento contiene una caja `<div id="secondary" class="sidebar-container" role="complementary">`. Esta, a su vez, contiene dos cajas `<div>`: una para los widgets (`<div class="widget-area masonry">`) y otra para el uso de WordPress (`<div class="site-info">`).

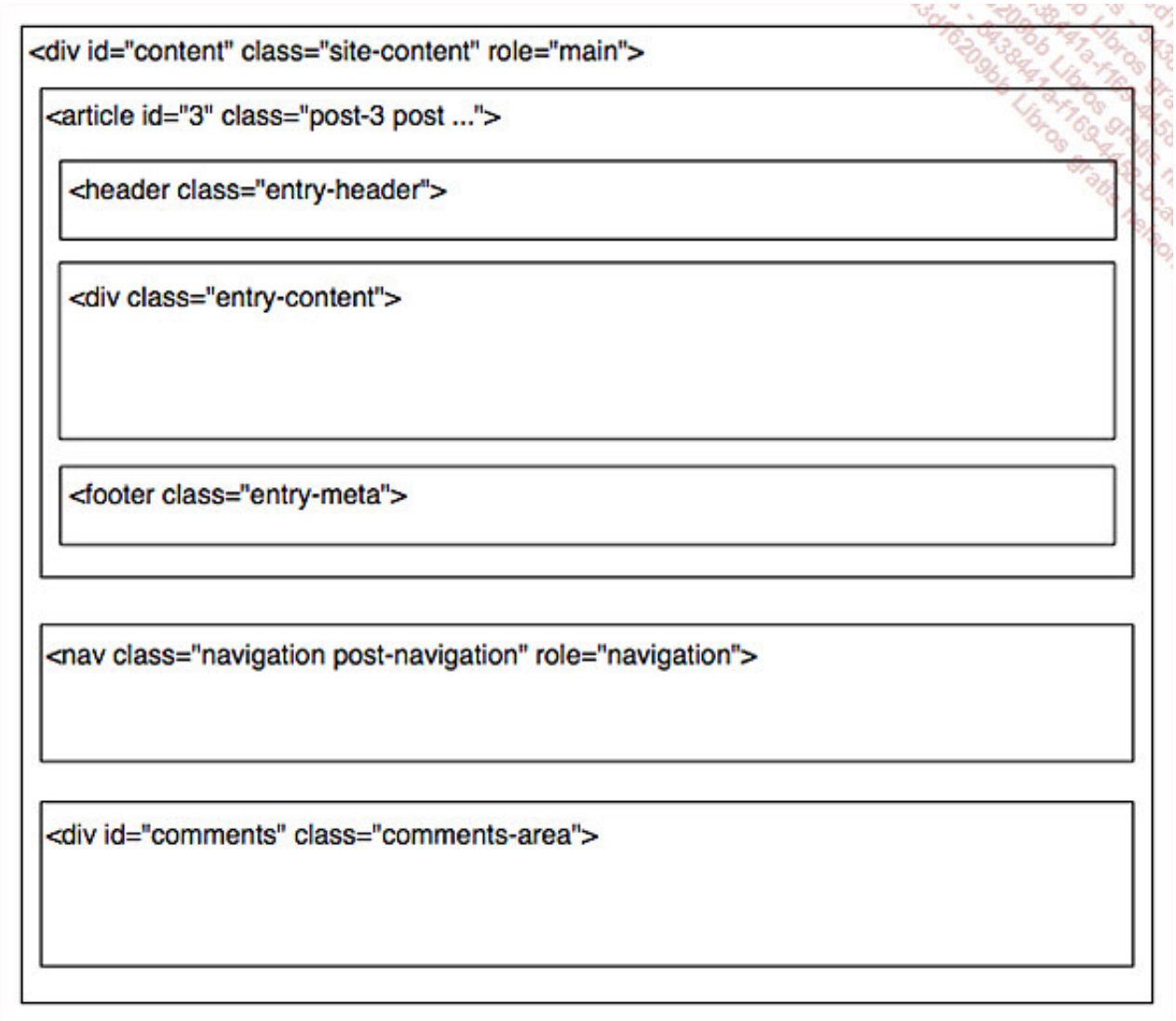
La caja de los widgets `<div class="widget-area masonry">` contiene tantos elementos`<aside>` como widgets ha situado el administrador del sitio.

La caja `<div class="site-info">` contiene únicamente un elemento de vínculo `<a>`.

5. Las entradas solas

Cuando un visitante hace clic en el título de una entrada de la página de inicio, la entrada se muestra en una página sola, con el formulario de los comentarios.

Esta es la estructura de una entrada sola:



Las entradas se sitúan en cajas `<div>` con un identificador único y numerosas clases que dependen de las características de publicación de cada entrada: `<article id="post-6" class="post-6 post type-post status-publish format-standard hentry category-uncategorized">`.

Las entradas cuentan con un encabezado: `<header class="entry-header">` que contiene el título de la entrada (en un `<h1>`) y los metadatos (en una caja `<div>`).

El contenido de redacción de la entrada se inserta en una caja `<div class="entry-content">`.

De acuerdo con las propiedades de cada entrada, es posible que se produzca el uso de metadatos. Estos se situarían en el elemento `<footer class="entry-meta">`.

A continuación encontramos elementos de navegación que permiten pasar rápidamente de una entrada a otra. Se trata del elemento `<nav class="navigation post-navigation" role="navigation">` que también posee un `role` definido.

Por último, el cuadro de comentarios y del formulario se sitúa en la caja `<div id="comments" class="comments-area">`.

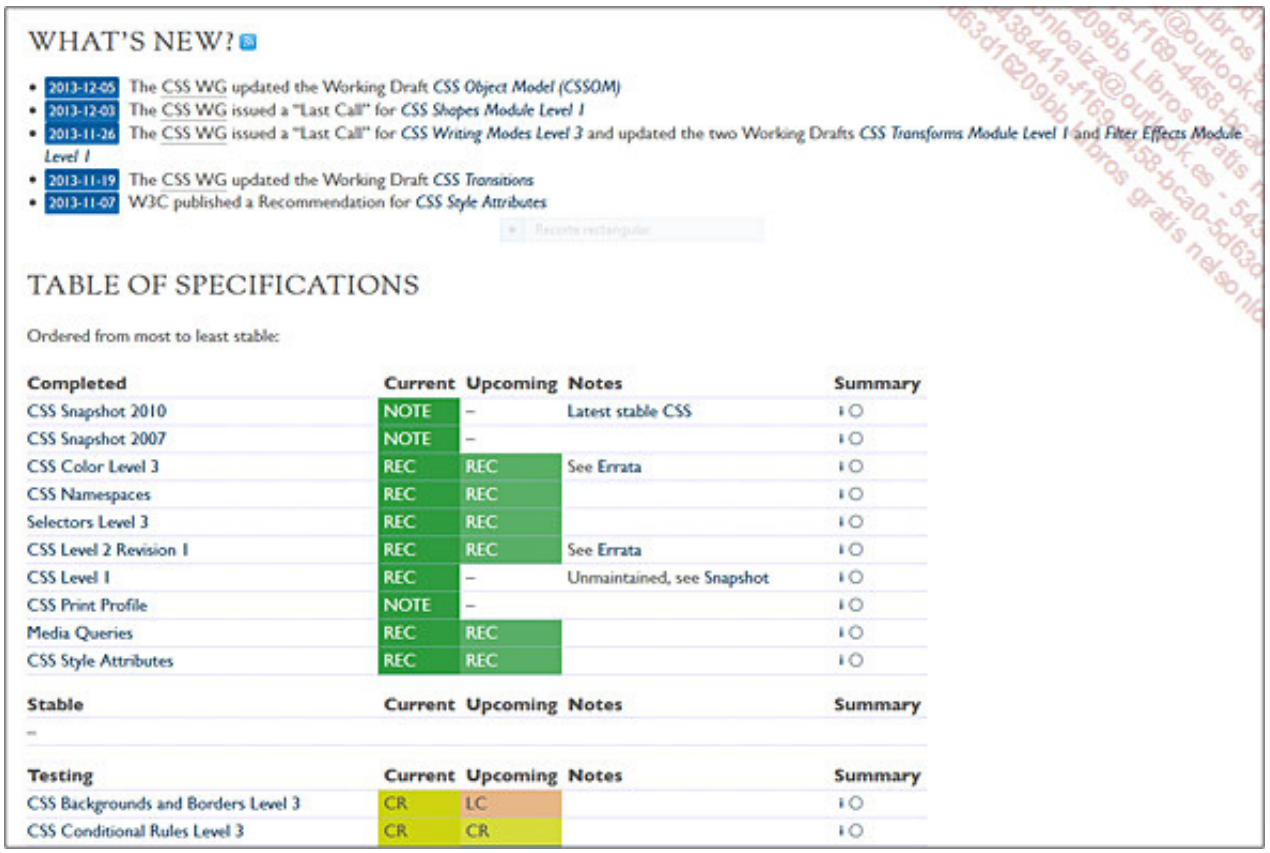
Análisis de la situación

1. Los módulos en curso

Por supuesto, el W3C ya está trabajando en el CSS3. El objetivo del W3C no es crear y redactar una única recomendación, sino trabajar en multitud de módulos que permitan organizar las propiedades CSS. Así, en lugar de crear una única y "gigantesca" especificación, resulta más productivo y más rápido avanzar con varias especificaciones independientes.

Para mantenerse informado de la evolución de los módulos CSS3, vaya a la siguiente URL del W3C:http://www.w3.org/Style/CSS/current-work

Este es un pequeño extracto (diciembre de 2013):



WHAT'S NEW?

- 2013-12-05 The CSS WG updated the Working Draft *CSS Object Model (CSSOM)*
- 2013-12-03 The CSS WG issued a "Last Call" for *CSS Shapes Module Level 1*
- 2013-11-26 The CSS WG issued a "Last Call" for *CSS Writing Modes Level 3* and updated the two Working Drafts *CSS Transforms Module Level 1* and *Filter Effects Module Level 1*
- 2013-11-19 The CSS WG updated the Working Draft *CSS Transitions*
- 2013-11-07 W3C published a Recommendation for *CSS Style Attributes*

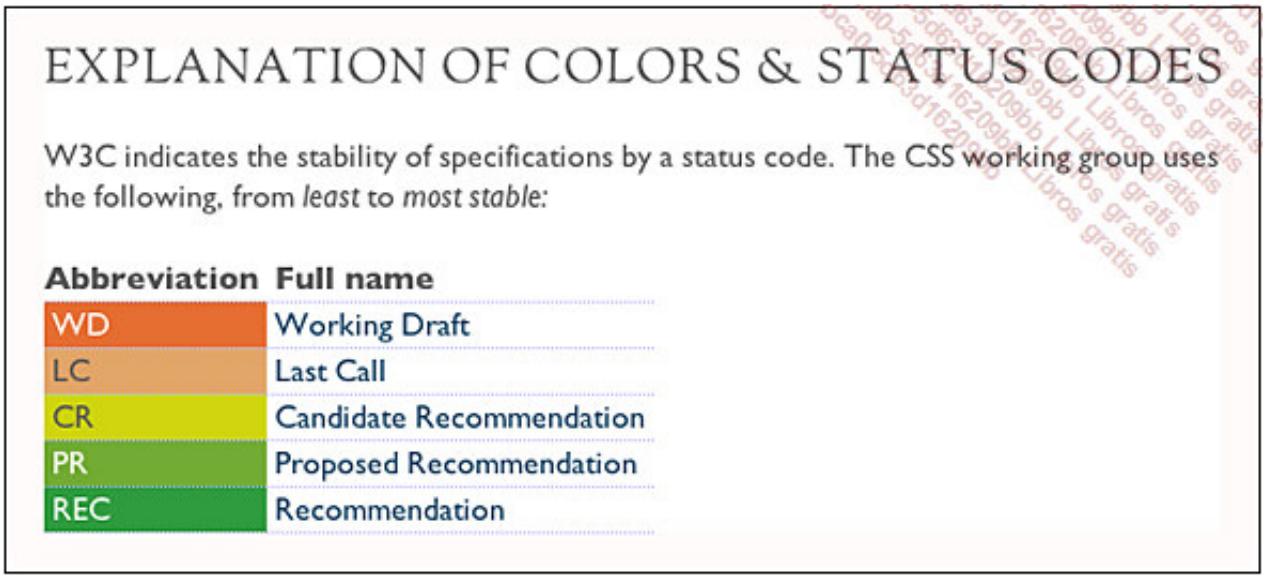
TABLE OF SPECIFICATIONS

Ordered from most to least stable:

Completed	Current	Upcoming	Notes	Summary
CSS Snapshot 2010	NOTE	–	Latest stable CSS	
CSS Snapshot 2007	NOTE	–		
CSS Color Level 3	REC	REC	See Errata	
CSS Namespaces	REC	REC		
Selectors Level 3	REC	REC		
CSS Level 2 Revision 1	REC	REC	See Errata	
CSS Level 1	REC	–	Unmaintained, see Snapshot	
CSS Print Profile	NOTE	–		
Media Queries	REC	REC		
CSS Style Attributes	REC	REC		
Stable	**Current**	**Upcoming**	**Notes**	**Summary**
–				
Testing	**Current**	**Upcoming**	**Notes**	**Summary**
CSS Backgrounds and Borders Level 3	CR	LC		
CSS Conditional Rules Level 3	CR	CR		

Usted podrá seguir la evolución de la redacción con las casillas de estado actual **Status**:

- **REC**: *Recommendation*. El módulo está oficialmente terminado y se propone como Recomendación.
- **PR**: *Proposed Recommendation*. El módulo está técnicamente finalizado y se ha enviado su propuesta al comité del W3C.
- **CR**: *Candidate Recommendation*. El módulo cumple con todos los requisitos según el grupo de trabajo y se ha estudiado en profundidad su implementación.
- **LC**: *Last Call*. Aún se está trabajando en el módulo, pero se ha presentado oficialmente ante los otros grupos de trabajo del W3C y el público en general.
- **WD**: *Working Draft*. El módulo se está elaborando, pero se pone a la disposición de los miembros del W3C y del público en general para su estudio técnico.



EXPLANATION OF COLORS & STATUS CODES

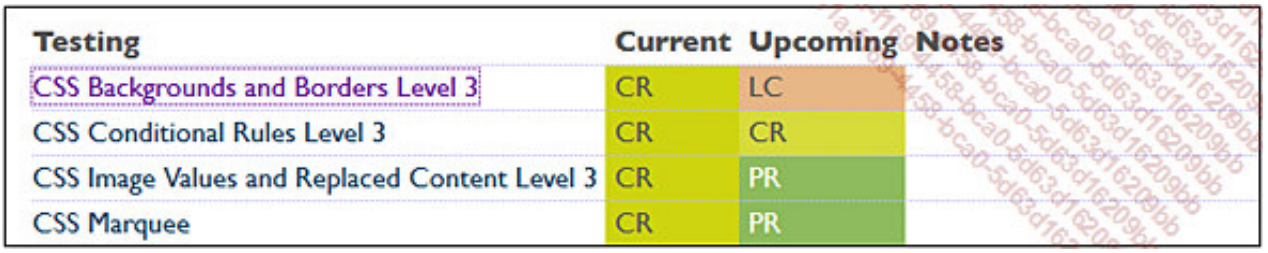
W3C indicates the stability of specifications by a status code. The CSS working group uses the following, from *least* to *most stable*:

Abbreviation	Full name
WD	Working Draft
LC	Last Call
CR	Candidate Recommendation
PR	Proposed Recommendation
REC	Recommendation

2. Consultar las especificaciones

Es posible acceder a los distintos módulos mediante el vínculo que indica su situación actual. Usted deberá consultar con regularidad los avances realizados para mantenerse informado de las novedades, las correcciones, las cancelaciones, las precisiones...

Veamos por ejemplo el CSS que hace referencia a los fondos y a los bordes. En la lista anterior de módulos, haga clic en el vínculo **CSS Backgrounds and Borders Level 3**.



Testing	Current	Upcoming	Notes
CSS Backgrounds and Borders Level 3	CR	LC	
CSS Conditional Rules Level 3	CR	CR	
CSS Image Values and Replaced Content Level 3	CR	PR	
CSS Marquee	CR	PR	

La página siguiente muestra el contenido del módulo de acuerdo con la evolución de los trabajos.

W3C Candidate Recommendation

CSS Backgrounds and Borders Module Level 3

W3C Candidate Recommendation 24 July 2012

This version:

http://www.w3.org/TR/2012/CR-css3-background-20120724/

Latest version:

http://www.w3.org/TR/css3-background/

Latest editor's draft:

http://dev.w3.org/csswg/css3-background/

Previous version:

http://www.w3.org/TR/2012/CR-css3-background-20120417/

Issue Tracking:

http://www.w3.org/Style/CSS/Tracker/products/11

Feedback:

www-style@w3.org with subject line "`[css3-background] … message topic …`"

Editors:

Bert Bos (W3C)

Elika J. Etemad (Mozilla)

Brad Kemper (Invited Expert)

Los prefijos para los navegadores

1. Las especificaciones y las recomendaciones

¿Podemos usar CSS3 desde ya? El problema siempre es el mismo: el W3C propone las especificaciones, pero son los navegadores quienes tienen la última palabra. Son los navegadores quienes deben estar informados de las novedades y quienes deben implementarlas en su motor de visualización. Para los navegadores, seguir día a día las novedades es una auténtica carrera contra reloj, es una misión imposible.

2. Los navegadores y el CSS3

Para evitar el estancamiento, el W3C ha encontrado una solución alternativa. Este le deja total libertad a los navegadores a la hora de implementar las novedades gracias a un prefijo único que especifica qué navegador se debe usar. Cuando la especificación alcanza el nivel **CR**, *Candidate Recommendation*, los prefijos ya no son necesarios.

Estos son los prefijos de los navegadores (conocidos como prefijos propietarios):

- `-moz-`: para el motor de renderizado Gecko de **Mozilla Firefox**.
- `-webkit-`: para el motor de renderizado WebKit de **Safari** y **Chrome**.
- `-o-`: para el motor de renderizado de **Opera**.
- `-ms-`: para el motor de renderizado de Microsoft **Internet Explorer**.
- `-khtml-`: para el motor de renderizado **KHTML** usado por varios navegadores en Linux.

Veamos un ejemplo concreto. Supongamos que queremos usar la propiedad `border-radius` que permite crear esquinas redondeadas para las cajas `<div>`.

Esta sería la sintaxis necesaria para que sea reconocido en "todos" los navegadores:

```
header {
    -moz-border-radius: 10px;
    -webkit-border-radius: 10px;
    -o-border-radius: 10px;
    -ms-border-radius: 10px;
    -khtml-border-radius: 10px;
    border-radius: 10px;
}
```

Las primeras líneas son específicas para cada motor de visualización mencionado anteriormente. La última línea corresponde a la propiedad estándar para todos los navegadores, quienes reconocerán la propiedad "más adelante", en cuanto el W3C la haya finalizado.

El orden de las líneas es importante. Al colocar la propiedad "estándar" en último lugar, nos aseguramos de que tendrá prioridad sobre las líneas precedentes. Por lo tanto, deberá indicar primero las propiedades con prefijos, que puede que cambien, y colocar la propiedad "oficial" al final.

Esta sintaxis puede parecer "pesada", pero se trata de la mejor solución, ya que ofrece la mejor portabilidad de las propiedades en espera de la oficialización.

3. Los generadores de prefijos en línea

Para los diseñadores web, escribir multitud de veces esas líneas resulta fastidioso. Para evitarlo, existen numerosos servicios en línea que permiten añadir los prefijos de las propiedades: los

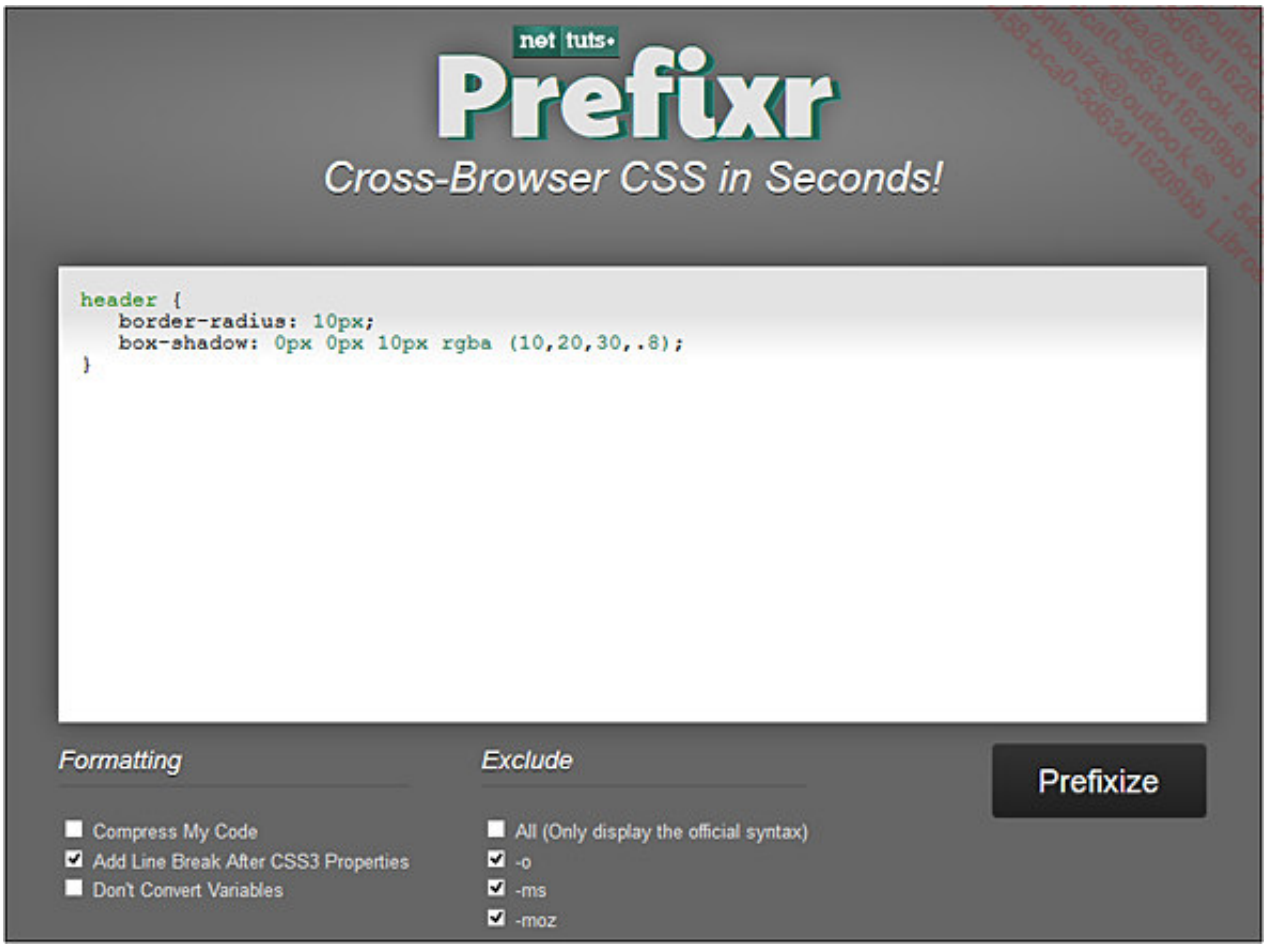
"generadores de prefijos".

En la interfaz de la herramienta, simplemente deberá introducir la propiedad o propiedades deseadas y el generador se encargará de crear toda la sintaxis.

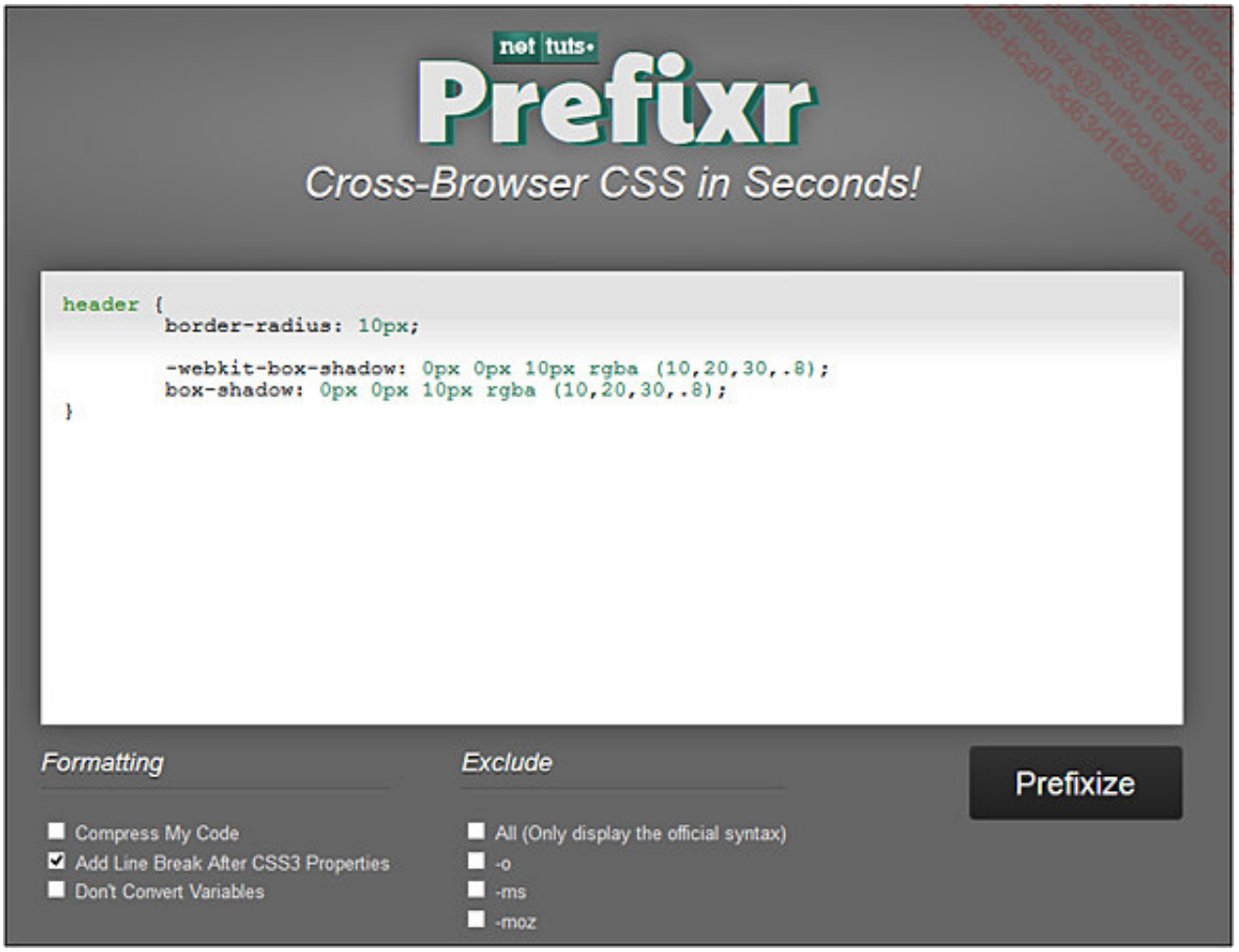
```
header {
    border-radius: 10px;
    box-shadow: 0px 0px 10px rgba(10,20,30,.8);
}
```

4. Net Tuts Prefixr

Este sitio web (http://prefixr.com/) le propone una interfaz muy fácil de usar. Introduzca su código CSS en el campo de texto, indique si desea excluir determinados prefijos y seleccione las opciones de formato.



Haga clic en el botón **Prefixize**. Los prefijos se añadirán automáticamente.



5. -prefix-my CSS

Este servicio (http://prefixmycss.com/) funciona de forma similar. En la parte inferior de la pantalla se indica la lista de las propiedades CSS3 admitidas. En el recuadro de la izquierda, **1. Paste your CSS**, indique sus reglas CSS y luego haga clic en el botón **Prefix!**. Las reglas CSS con los prefijos aparecerán en el recuadro **3.Get your -prefixed-CSS**.

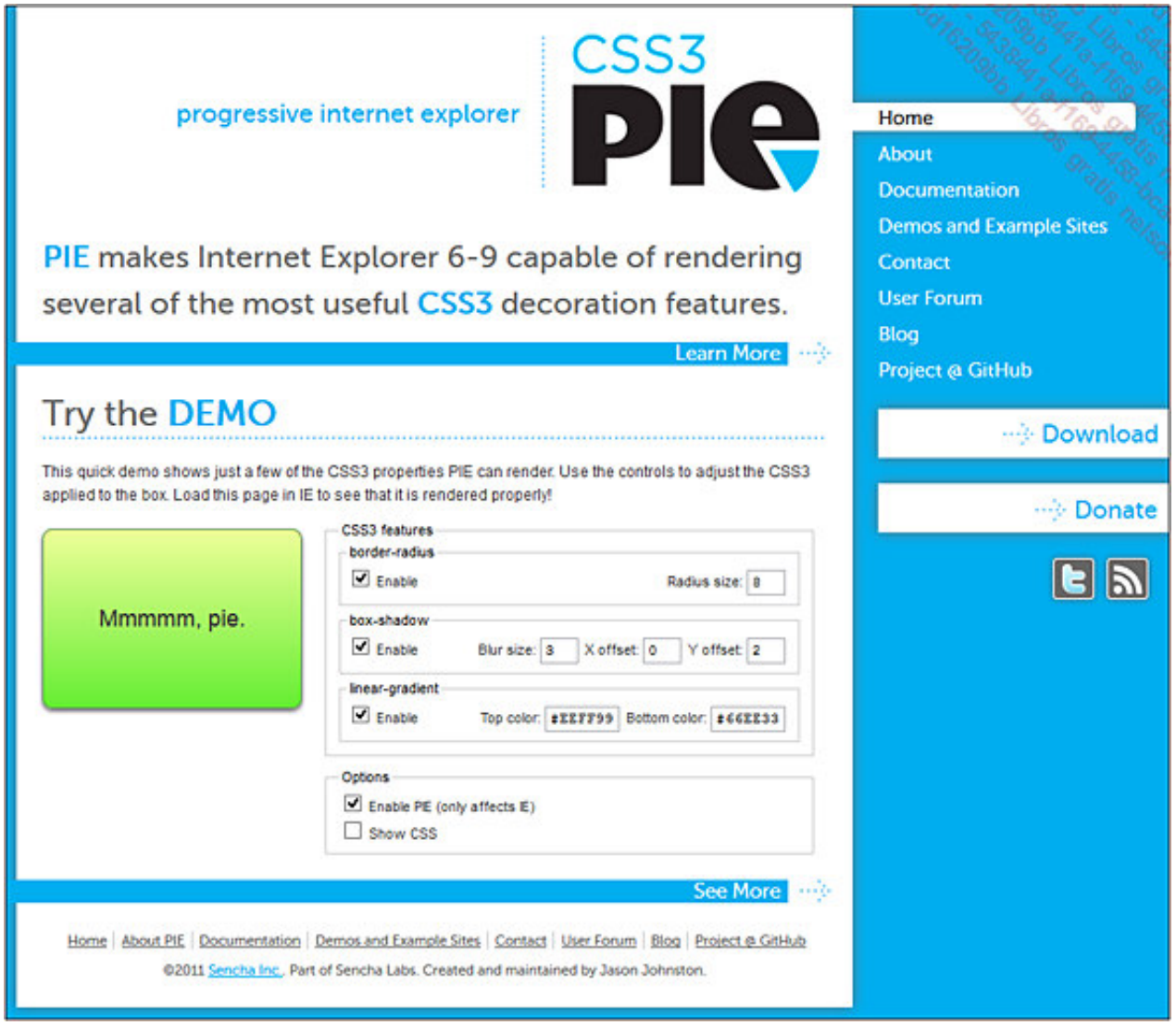
6. CSSPrefixer

Este sitio web (http://cssprefixer.appspot.com/) le propone una interfaz minimalista: usted deberá indicar sus reglas y hacer clic en el botón **process** para añadir los prefijos. No hay opciones.

7. El caso de Internet Explorer

Tendremos problemas, como siempre, con el navegador de Microsoft, en espera de que llegue la prometedora versión 10 y sea adoptada por la mayoría de usuarios de Windows. Las versiones anteriores no reconocen multitud de reglas CSS3.

El sitio web **CSS3 PIE** (http://css3pie.com/) le permite descargar un script para adaptar determinadas propiedades como `border-radius`, `box-shadow`, `border-image`... de modo que Internet Explorer sea capaz de "digerirlas".



8. Un generador de prefijos JavaScript

Esta solución (http://leaverou.github.io/prefixfree/) le permite vincular sus páginas HTML a un archivo JavaScript muy ligero (de a penas 2Ko), que generará automáticamente los prefijos en función del navegador que use el visitante de su página web. ¡No será necesario insertar los prefijos, ni tendrá que conocerlos o mantenerse informado de su evolución!

Este script añade una clase al elemento `<html>`, con el valor del prefijo propietario del navegador en cuestión.

Esta solución funciona con los navegadores siguientes: Internet Explorer 9 y versiones superiores, Opera 10 y versiones superiores, Firefox 3.5 y versiones superiores, Safari 4 y versiones superiores y Chrome. Para las versiones móviles, prefix-free funciona con Mobile Safari, Android Browser, Chrome y Opera Mobile.

¡Simplemente genial! Lea con atención las limitaciones.

Haga clic en el botón **Only 2KB gzipped**, en el margen superior izquierdo (¡podrá ver de paso una magnífica animación CSS!). Haga **Guardar como** en el archivo JavaScript y guárdelo con el nombre**prefixfree.min.js**.

Cree una página HTML5 con un elemento, solamente para hacer el test.

```
<!DOCTYPE HTML>
<html lang="es">
<head>
<title>Mi documento</title>
<meta charset="UTF-8" />
<style>

</style>
</head>
<body>
<header>
    <h1>El título de mi página web</h1>
    <p>Cras mattis...</p>
</header>
</body>
</html>
```

Inserte ahora las reglas CSS3, sin prefijos para los navegadores.

```
<style>
body {
    font: .8em Verdana, Arial, Helvetica, sans-serif;
}
h1, p {
    margin: 0;
}
header {
    width: 400px;
    padding: 10px;
    border-radius: 10px;
    box-shadow: 0px 0px 10px rgba(10,20,30,.8);
    background-image: linear-gradient(90deg, transparent,
    rgba(10,0,0,.3)),linear-gradient(transparent, transparent);
```

```
    box-sizing: border-box;
    cursor: zoom-in;
    }
</style>
```

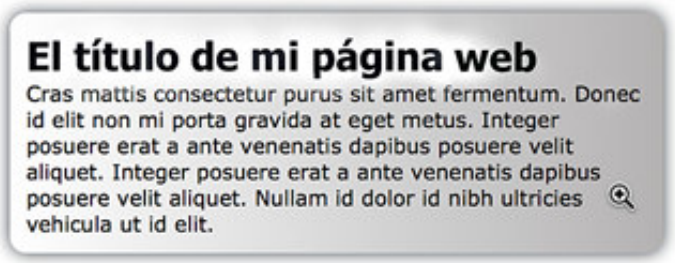
También puede crear un archivo .css y vincularlo a su página HTML:

```
<link href="mi-hoja.css" rel="stylesheet" />
```

La programadora Lea Verou recomienda crear el vínculo al archivo JavaScript debajo de la línea de estilos:

```
<style>
   ...
</style>
<script src="prefixfree.min.js" type="text/javascript"></script>  </head>
```

Publique la página en Internet y visualice el código fuente:



Para los navegadores que necesitan prefijos para las reglas CSS utilizadas, el JavaScript modificará el código CSS del archivo. Los navegadores que no necesitan modificaciones permanecerán sin cambios.

Este es un extracto de código generado con **Safari 6**:

Para el encabezado `<html>`:

```
<html lang="es" class="-webkit-">
```

Para el estilo CSS `header`, tenemos el añadido del prefijo `-webkit-` :

```
header {
   width: 400px;
   padding: 10px;
   border-radius: 10px;
   box-shadow: 0px 0px 10px rgba(10,20,30,.8);
   background-image: -webkit-linear-gradient(0deg, transparent,
rgba(10,0,0,.3)),-webkit-linear-gradient(transparent, transparent);
   box-sizing: border-box;
   cursor: -webkit-zoom-in;
}
```

Este es un extracto del código generado con **Firefox 23**:

Para el encabezado `<html>`:

```
<html class=" -moz-" lang="es">
```

Para el estilo CSS `header`, tenemos el añadido del prefijo `-moz-`:

```
header {
   width: 400px;
```

```
   padding: 10px;
   border-radius: 10px;
   box-shadow: 0px 0px 10px rgba(10,20,30,.8);
   background-image: linear-gradient(90deg, transparent,
rgba(10,0,0,.3)),linear-gradient(transparent, transparent);
   -moz-box-sizing: border-box;
   cursor: -moz-zoom-in;
}
```

Cómo interpretan los navegadores el CSS3

1. ¿Qué propiedades podemos usar?

Como acabamos de ver, los módulos CSS3 del W3C están en constante evolución. Los navegadores web intentan "seguir el movimiento" tratando de reconocer y de interpretar las nuevas propiedades CSS3.

Para facilitarle las cosas, dispone de multitud de servicios en línea.

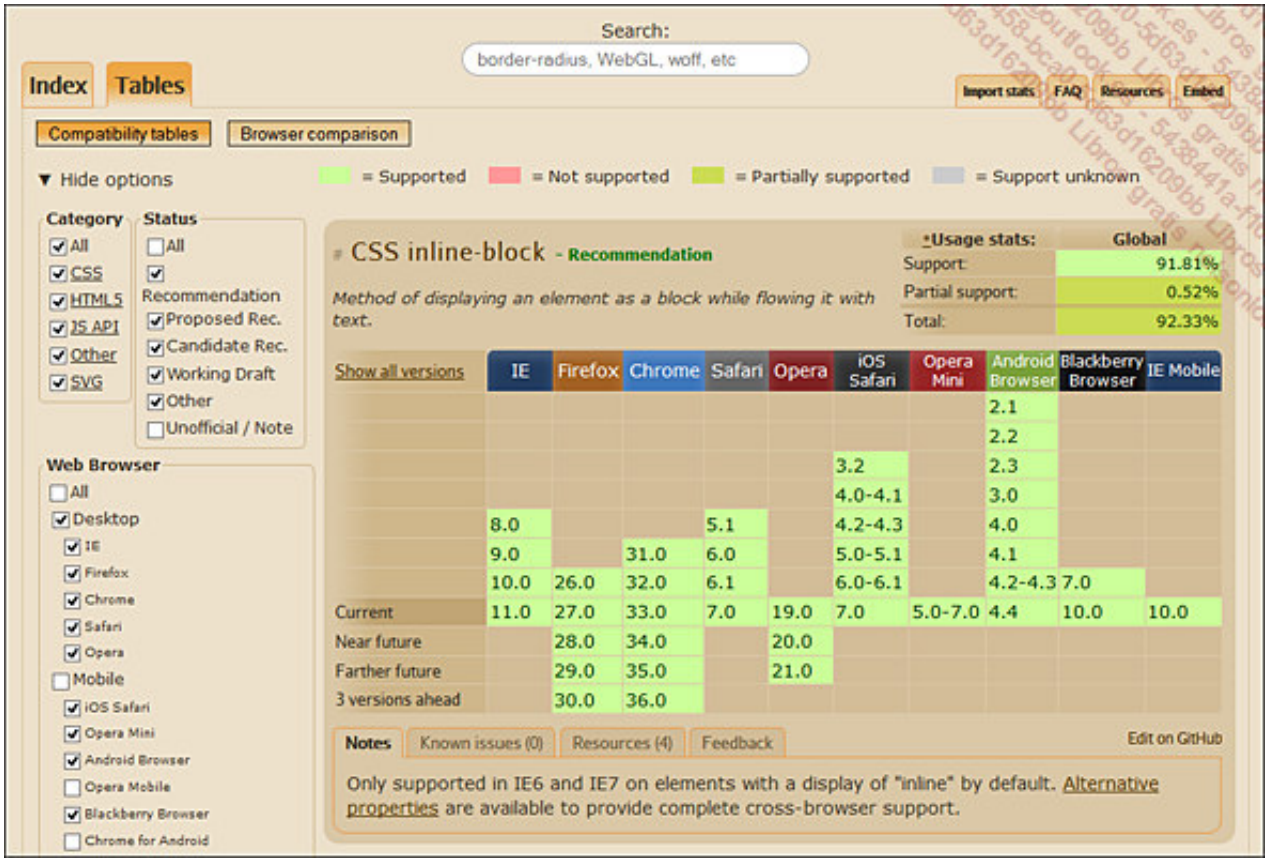
2. findmebyIP - HTML5 & CSS3 Support

Este sitio web (http://www.findmebyip.com/litmus/) le indica la compatibilidad con los principales navegadores del mercado de las propiedades CSS3 (y de los elementos de HTML5), así como de los selectores.

3. When can I use

El sitio web **When can I use** (http://caniuse.com/), como su nombre indica, propone un servicio muy completo sobre el grado de compatibilidad de los navegadores con las propiedades CSS3.

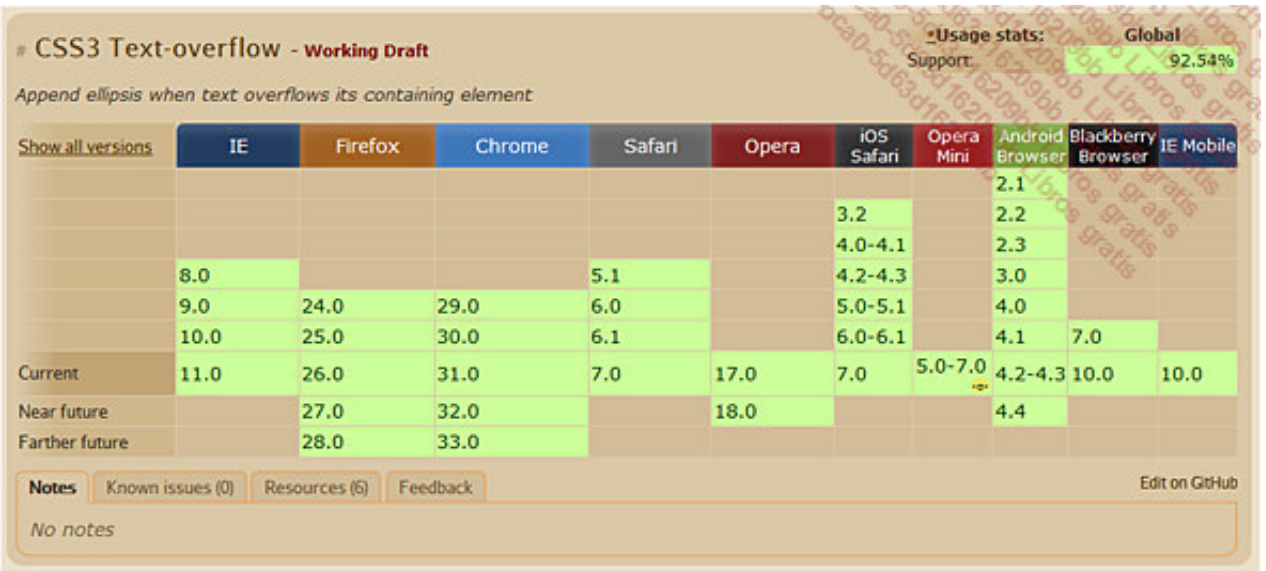
Si hace clic en **Tables** y, a continuación, en **Show options**, podrá ordenar la lista en función de multitud de criterios.



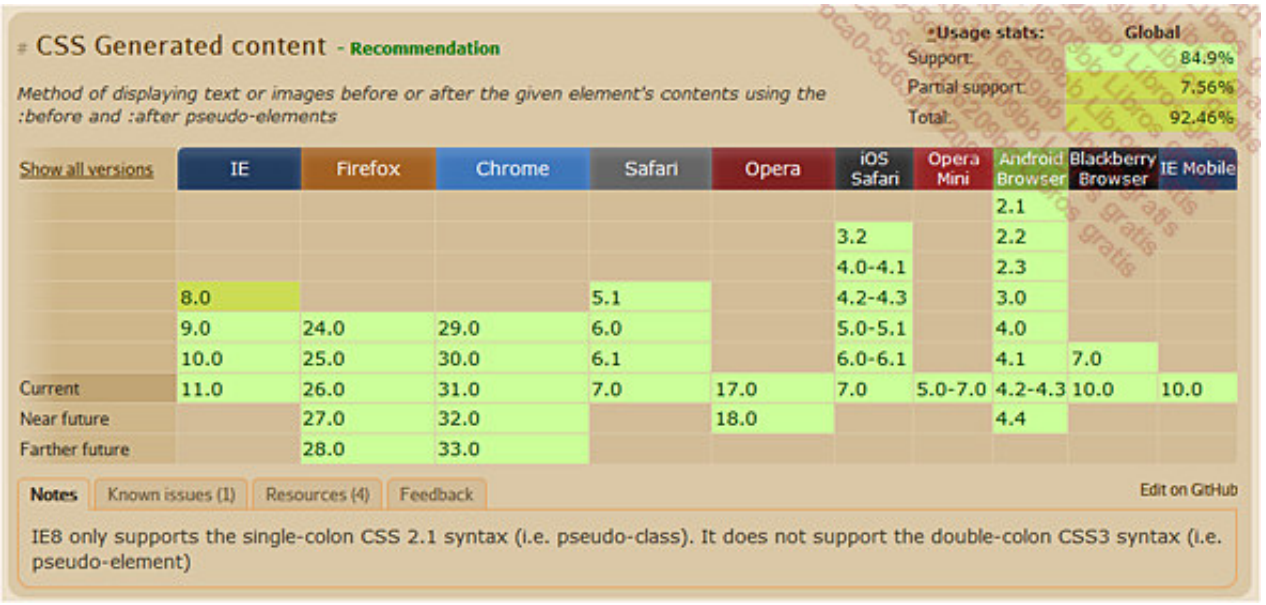
Un código de colores le indicará el grado de compatibilidad:

En cuanto a las propiedades, el sitio web le indica el estado actual del módulo: **Working Draft**,**Candidate recommendation**, **Proposed Recommendation** o **Recommendation**.

En este ejemplo, la propiedad **CSS3 Text-overflow** tiene el estatus de **Working Draft**.



Para algunas propiedades, podrá encontrar recursos adicionales bajo las diferentes pestañas:**Notes**, **Known issues**, **Resources** y **Feedback**.



CSS Generated content - Recommendation

*Usage stats: Global

Support: 84.9%

Partial support: 7.56%

Total: 92.46%

Method of displaying text or images before or after the given element's contents using the :before and :after pseudo-elements

Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	IE Mobile
								2.1		
						3.2		2.2		
						4.0-4.1		2.3		
	8.0			5.1		4.2-4.3		3.0		
	9.0	24.0	29.0	6.0		5.0-5.1		4.0		
	10.0	25.0	30.0	6.1		6.0-6.1		4.1	7.0	
Current	11.0	26.0	31.0	7.0	17.0	7.0	5.0-7.0	4.2-4.3	10.0	10.0
Near future		27.0	32.0		18.0			4.4		
Farther future		28.0	33.0							

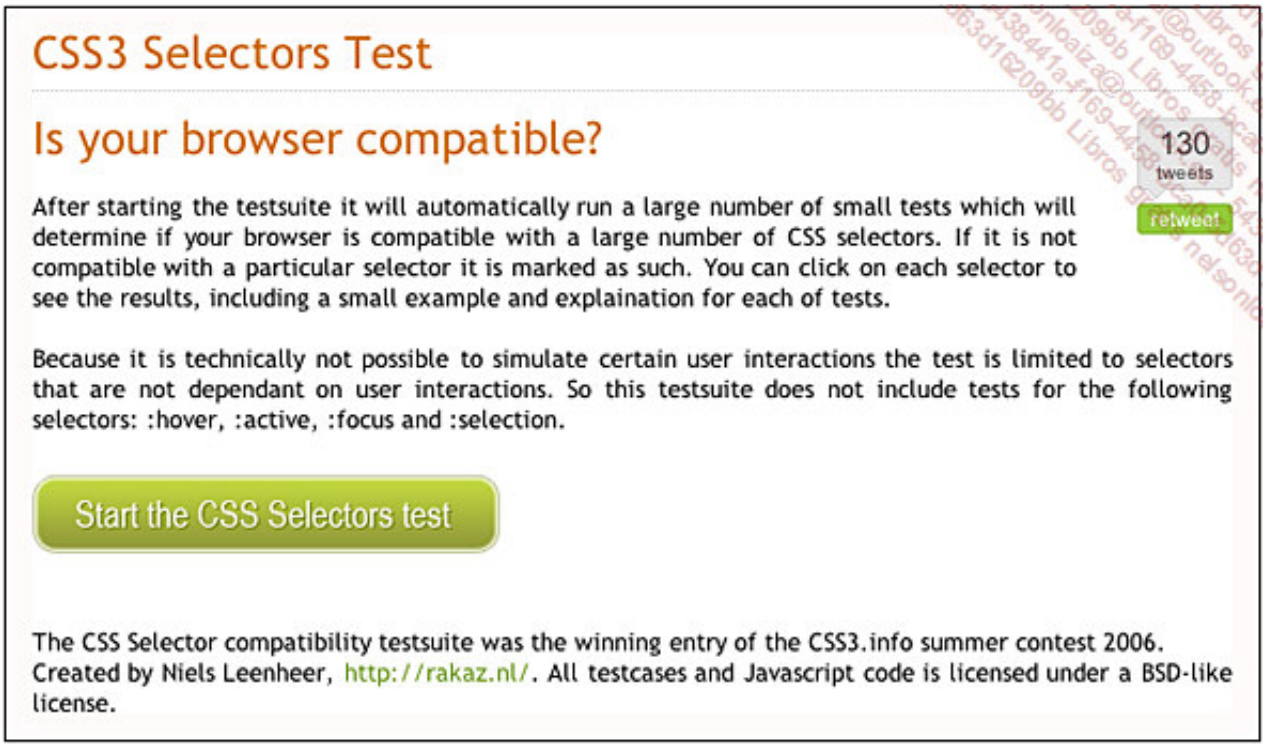
Notes | Known issues (1) | Resources (4) | Feedback | Edit on GitHub

IE8 only supports the single-colon CSS 2.1 syntax (i.e. pseudo-class). It does not support the double-colon CSS3 syntax (i.e. pseudo-element)

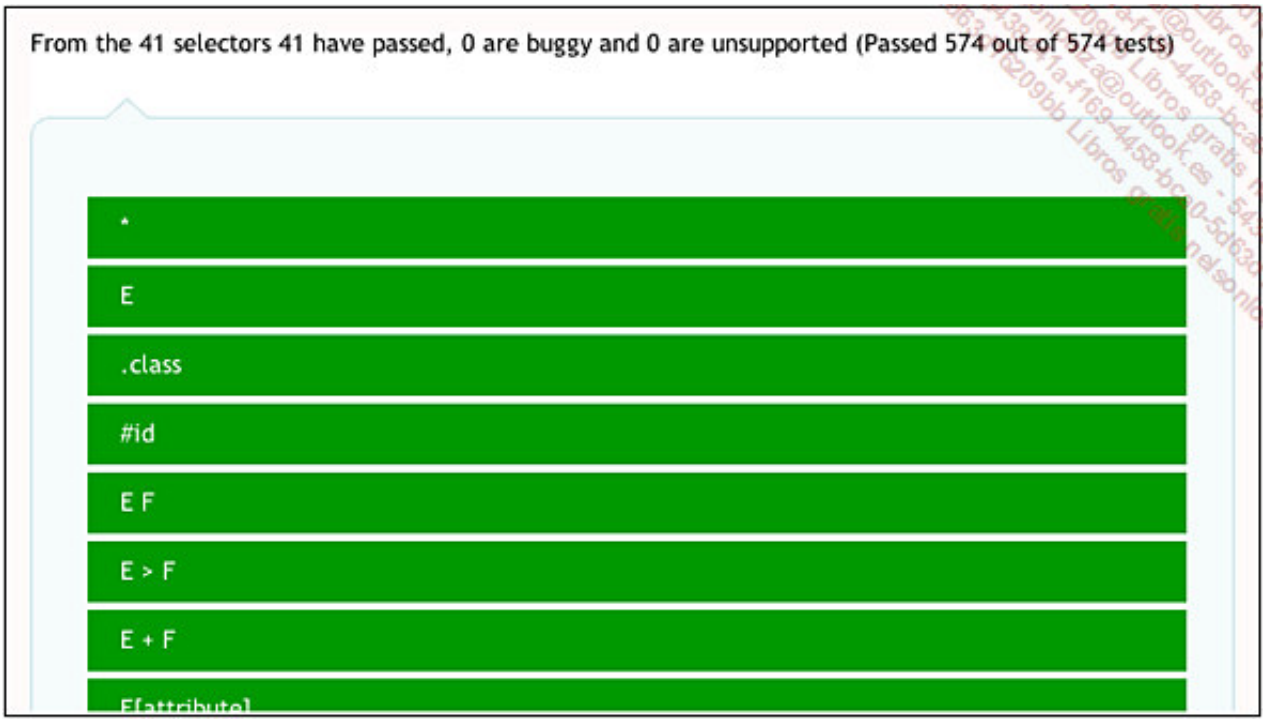
4. Examinar la compatibilidad con los selectores

El sitio web CSS3.info (http://www.css3.info/) le permite examinar el grado de compatibilidad de su navegador con los selectores CSS3.

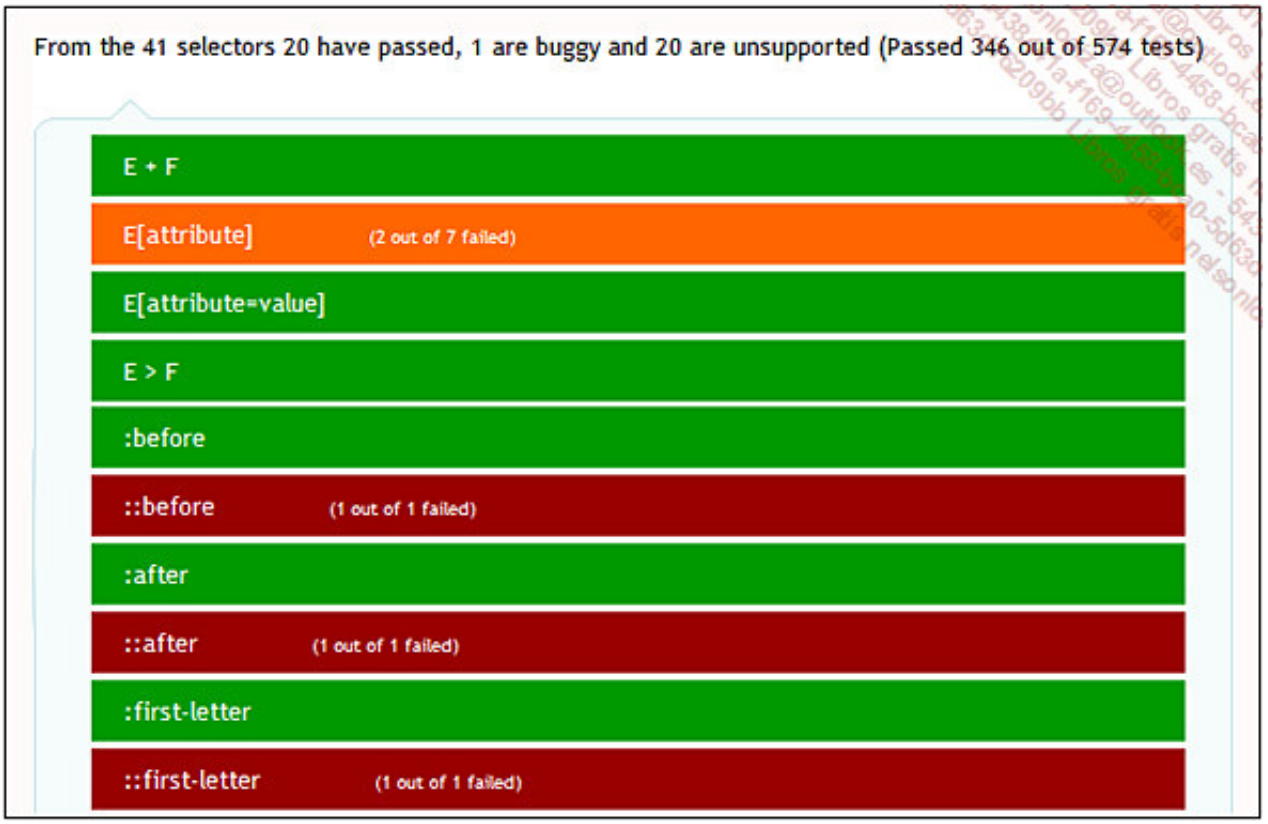
Abra un navegador y vaya a la URL: http://www.css3.info/selectors-test/. Haga clic en el botón **Start the CSS Selectors test**.



El test comenzará y se mostrarán los resultados: no ha encontrado ni el más mínimo error con un navegador reciente.



A continuación se indican los problemas encontrados con un navegador más antiguo:



5. CSS3 Please

Este sitio web (http://css3please.com/) es muy práctico.

Podemos consultar las principales propiedades CSS3, en la parte izquierda de la página, con los prefijos propietarios de los navegadores.

```
.box_shadow {
  -webkit-box-shadow: 0px 0px 4px #ffffff; /* Saf3-4 */
     -moz-box-shadow: 0px 0px 4px #ffffff; /* FF3.5 - 3.6 */
          box-shadow: 0px 0px 4px #ffffff; /* Opera 10.5, IE9, FF4+, Chrome 10+ */
}
```

Y, en la parte derecha, tenemos un ejemplo completo de una caja.

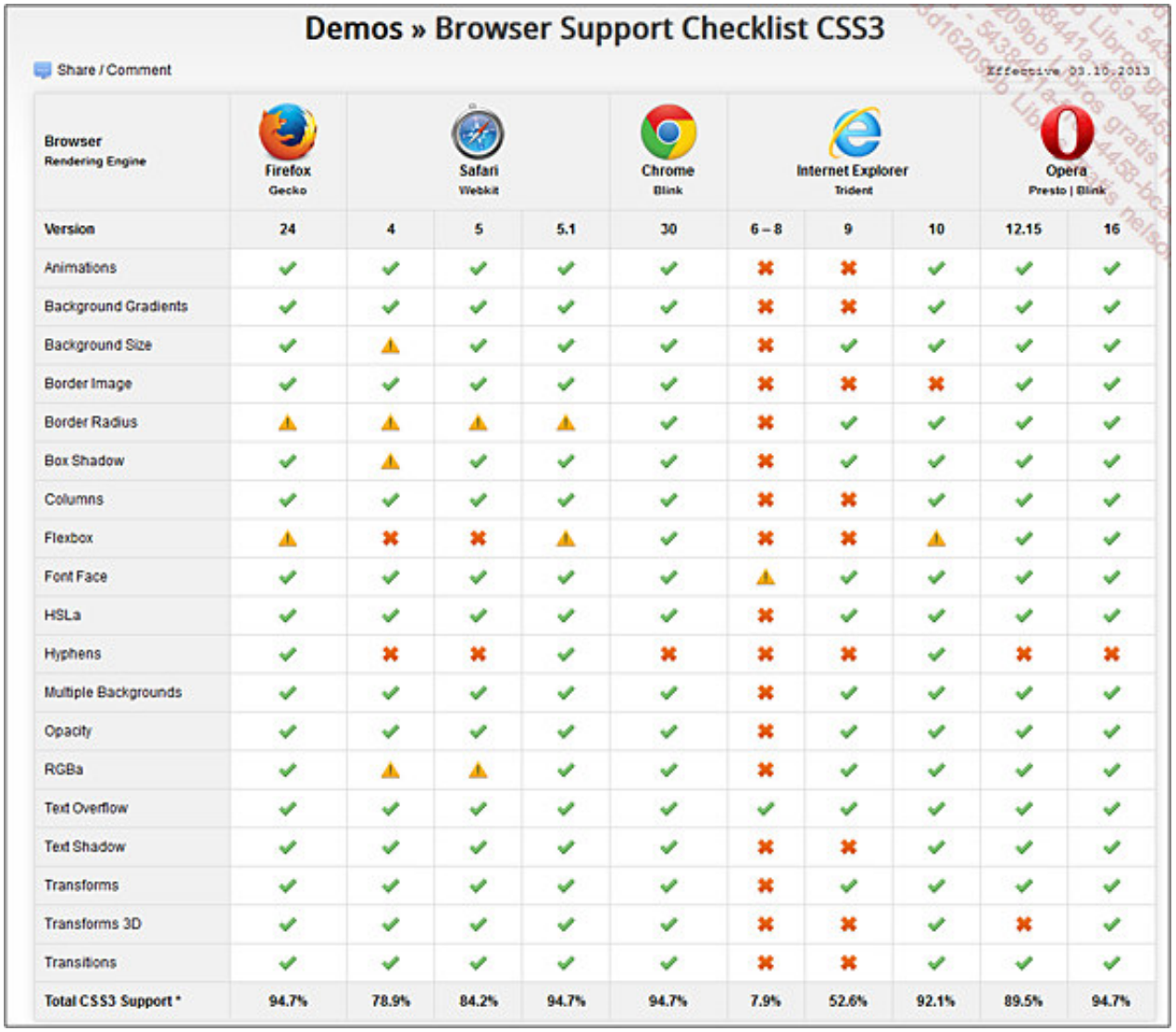
En la lista de la derecha podrá desactivar las distintas propiedades con el vínculo **[toggle rule off]**, y luego podrá volver a activarlas con **[toggle rule on]**, para ver si su navegador es capaz de interpretarlas y comprobar si se visualizan correctamente.

```
/*                          [to clipboard] [toggle rule off] */
.box_shadow {
  -webkit-box-shadow: 0px 0px 4px #ffffff; /* Saf3-4 */
     -moz-box-shadow: 0px 0px 4px #ffffff; /* FF3.5 - 3.6 */
          box-shadow: 0px 0px 4px #ffffff; /* Opera 10.5, IE9, FF4+, Chrome 10+ */
}
```

Además, podrá copiar el código de los ejemplos con el vínculo **[to clipboard]**.

6. Browser Support Checklist CSS3

Este sitio web (http://www.normansblog.de/demos/browser-support-checklist-css3/) presenta la lista de los elementos compatibles el 07/02/2012. Habrá que ver si el responsable del sitio web mantiene la tabla actualizada.



Demos » Browser Support Checklist CSS3

Share / Comment

Effective 03.10.2013

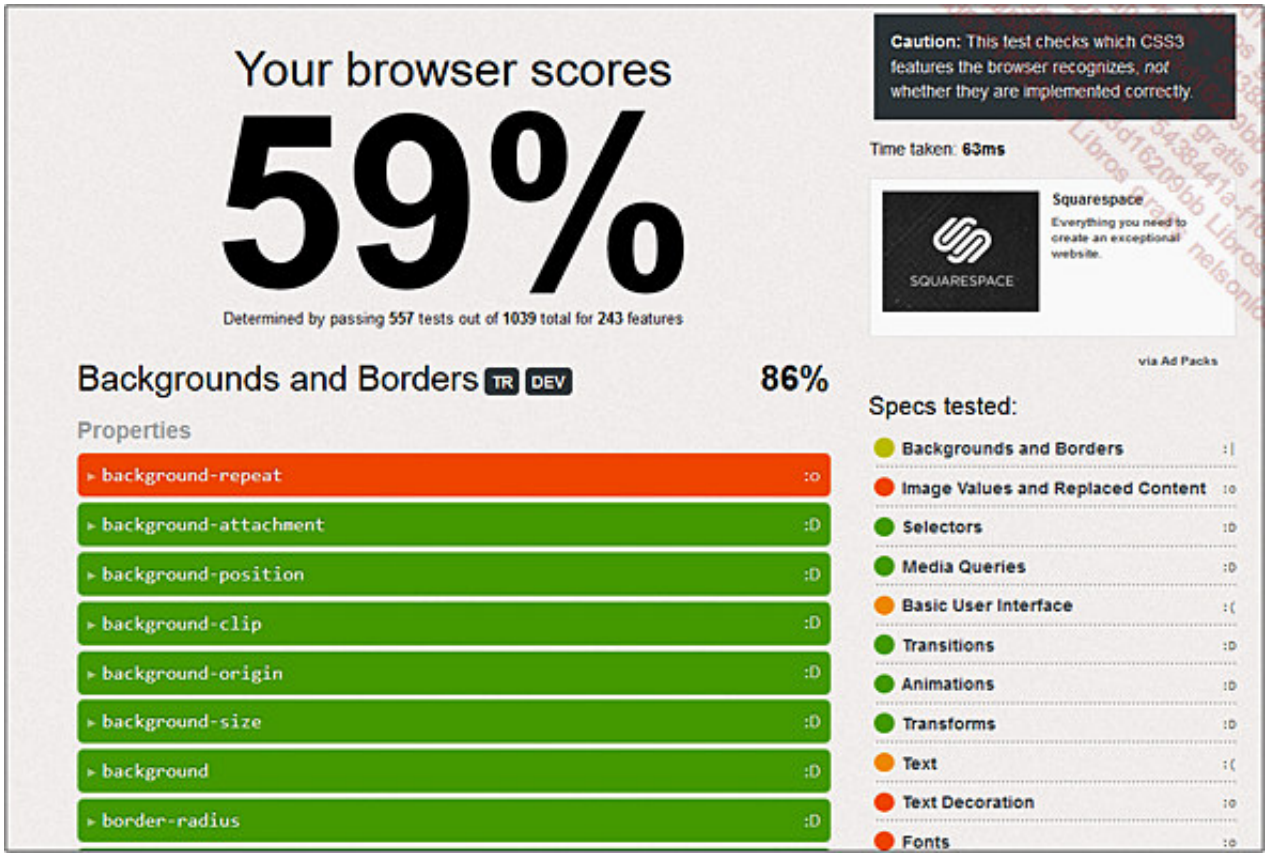
Browser Rendering Engine	Firefox Gecko	Safari Webkit			Chrome Blink	Internet Explorer Trident			Opera Presto | Blink	
Version	24	4	5	5.1	30	6 – 8	9	10	12.15	16
Animations	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓
Background Gradients	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓
Background Size	✓	⚠	✓	✓	✓	✗	✓	✓	✓	✓
Border Image	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓
Border Radius	⚠	⚠	⚠	⚠	✓	✗	✓	✓	✓	✓
Box Shadow	✓	⚠	✓	✓	✓	✗	✓	✓	✓	✓
Columns	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓
Flexbox	⚠	✗	✗	⚠	✓	✗	✗	⚠	✓	✓
Font Face	✓	✓	✓	✓	✓	⚠	✓	✓	✓	✓
HSLa	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
Hyphens	✓	✗	✗	✓	✗	✗	✗	✓	✗	✗
Multiple Backgrounds	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
Opacity	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
RGBa	✓	⚠	⚠	✓	✓	✗	✓	✓	✓	✓
Text Overflow	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Text Shadow	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓
Transforms	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
Transforms 3D	✓	✓	✓	✓	✓	✗	✗	✓	✗	✓
Transitions	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓
Total CSS3 Support *	94.7%	78.9%	84.2%	94.7%	94.7%	7.9%	52.6%	92.1%	89.5%	94.7%

7. css3test

Lea Verou (http://lea.verou.me) es una desarrolladora muy conocida en la comunidad Web. Propone numerosas herramientas (http://lea.verou.me/projects/) para ayudarnos a desarrollar nuestros sitios Web.

Su página css3test (http://css3test.com) permite comprobar si nuestro navegador reconoce las CSS3.

Este es el extracto de un test:


Your browser scores
59%
Determined by passing 557 tests out of 1039 total for 243 features
Caution: This test checks which CSS3 features the browser recognizes, not whether they are implemented correctly.
Time taken: 63ms
SQUARESPACE
Squarespace
Everything you need to create an exceptional website.
via Ad Packs
Backgrounds and Borders TR DEV 86%
Properties
background-repeat :o
background-attachment :D
background-position :D
background-clip :D
background-origin :D
background-size :D
background :D
border-radius :D
Specs tested:
Backgrounds and Borders :|
Image Values and Replaced Content :o
Selectors :D
Media Queries :D
Basic User Interface :(
Transitions :D
Animations :D
Transforms :D
Text :(
Text Decoration :o
Fonts :o

Los selectores CSS 2

1. Funcionalidad

Los selectores sirven para indicar a qué elementos se les van a aplicar los estilos definidos.

2. Los selectores

Debemos recordar que la recomendación CSS 2.1 ya proponía una amplia lista de selectores.

- **Selector universal**: permite aplicar un estilo a la totalidad de los elementos HTML de la página. Sintaxis: `* {...}`.
- **Selector de elemento o de tipo**: permite aplicar un estilo a un elemento HTML existente. Sintaxis: `p {...}`.
- **Selector de clase**: permite aplicar un estilo a los elementos que utilicen esa clase. Sintaxis:`.intro {...}`.
- **Selector de clase de un elemento**: permite aplicar un estilo a un elemento específico que utilice dicha clase. Sintaxis: `p.intro {...}`.
- **Selector de descendientes**: permite aplicar un estilo a un elemento determinado que se encuentre dentro de un elemento específico. Sintaxis: `p .nombrePropio {...}`.
- **Selector hijo**: permite aplicar un estilo al primer elemento hijo de un elemento específico. Sintaxis: `p>.nombrePropio {...}`.
- **Selector adyacente**: permite aplicar un estilo al elemento que se encuentre a continuación de un elemento específico. Sintaxis: `h1+h2 {...}`.
- **Selector de atributo**: permite aplicar un estilo a un elemento que utilice un atributo específico. Sintaxis: `p[lang] {...}`.
- **Selector de ID**: permite aplicar un estilo a un elemento que disponga de un código de identificación único. Sintaxis: `#bannerSuperior {...}`.

3. Las pseudo-clases

Las pseudo-clases se añaden al selector de manera que podamos aplicar un formato específico.

- La **pseudo-clase first-child** permite aplicar un estilo al primer elemento hijo de un elemento. Sintaxis: `ul li:first-child {...}`.
- Las **pseudo-clases de vínculos** permiten aplicar estilos a los vínculos en función de su estado (visitado y no visitado). Sintaxis: `a:link {...}` y `a:visited {...}`.
- Las **pseudo-clases dinámicas** permiten aplicar estilos a los vínculos cuando se pasa el ratón por encima o cuando se ha hecho clic en ellos, así como a los campos de los formularios que estén activos. Sintaxis: `a:hover {...}`, `a:active {...}` y `.campoTexto:focus {...}`.
- La **pseudo-clase de lenguaje** permite aplicar estilos en función del idioma especificado. Sintaxis: `:lang(es) {...}`.

4. Los pseudo-elementos

Los pseudo-elementos se añaden a un selector de manera que podamos aplicar un formato

específico.

- Los **pseudo-elementos de primera letra** y **de primera línea**. Con el pseudo-elemento de primera letra es posible aplicar un estilo a la primera letra de un elemento.

  Sintaxis: `.especial:first-letter {...}`.

  Se sigue el mismo principio para la primera línea.

  Sintaxis: `.especial:first-line {...}`.

- Los **pseudo-elementos antes y después**. Con estos pseudo-elementos es posible generar texto antes o después de un elemento determinado. Sintaxis: `.nota:before {...}` y `.destacado:after {...}`.

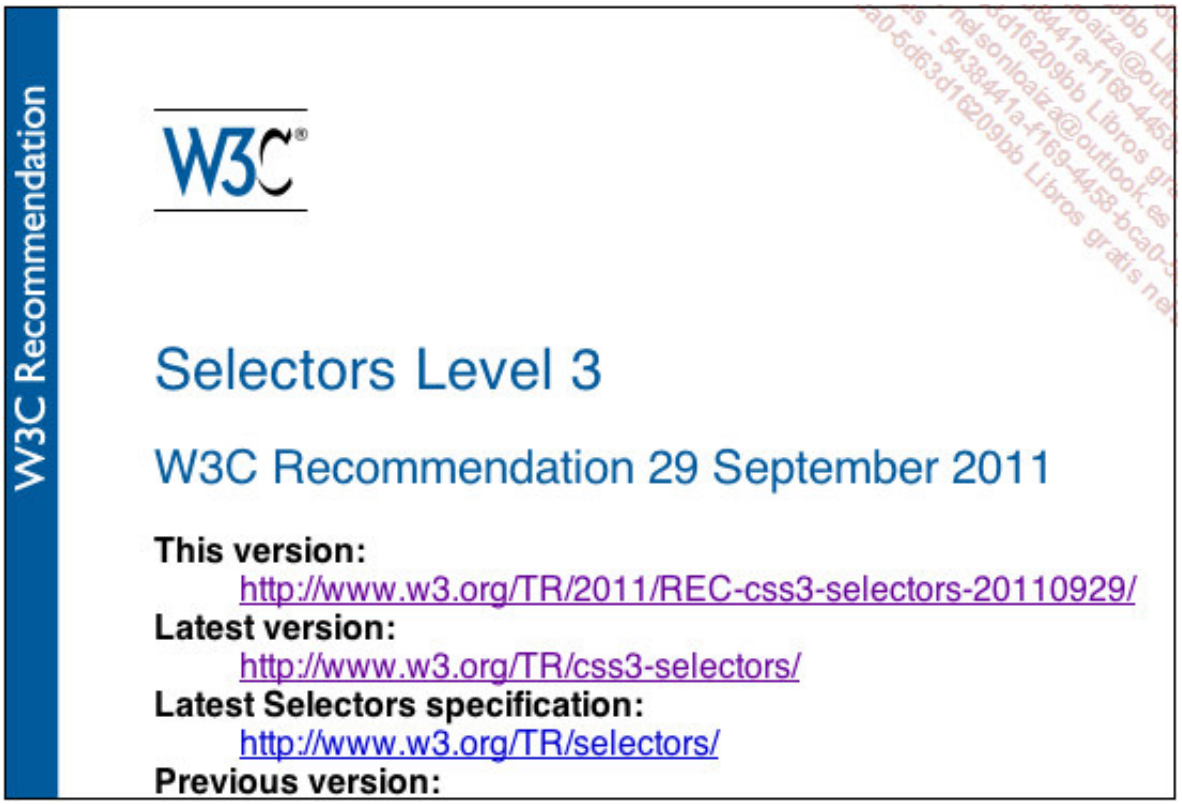
5. La agrupación de selectores

La agrupación de selectores permite aplicar propiedades comunes a diferentes selectores.

Sintaxis: `h1, .special, p.intro {...}`.

Los nuevos selectores CSS3

Las especificaciones CSS3 introducen nuevos selectores. El módulo de los selectores ya está terminado, así que el W3C ha propuesto una **Recommendation** con fecha del 29 de septiembre de 2011. Esta es su URL: http://www.w3.org/TR/css3-selectors/



El selector general de elementos adyacentes

El CSS2 nos proponía el selector adyacente, que permitía seleccionar el elemento hermano situado a continuación de otro elemento. Ejemplo de la sintaxis: `h1+h2 {...}`. De este modo solamente era posible aplicar un estilo específico al primer `h2` que siguiera al `h1`.

Con el CSS3, el elemento adyacente no tiene que estar obligatoriamente inmediatamente a continuación del primer elemento, pueden haber otros elementos entre ambos, y el estilo se aplicará a los elementos siguientes que se hayan especificado.

Esta es la sintaxis: `h1~h2 {...}`.

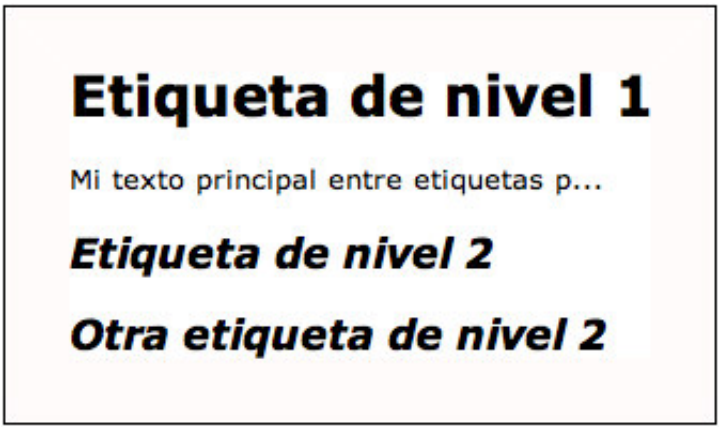
Ejemplo de selector:

```
h1~h2 {font-style: italic};
```

Ejemplo de código:

```
<h1>Etiqueta de nivel 1</h1>
<p>Mi texto principal entre etiquetas p...</p>
<h2>Etiqueta de nivel 2</h2>
<h2>Otra etiqueta de nivel 2</h2>
```

Y el resultado obtenido:



Como podemos ver, se ha aplicado la cursiva al texto de los dos elementos `<h2>` que se encuentran a continuación del `<h1>`, aunque tengamos un elemento `<p>` en medio.

El selector de atributo

1. Funcionalidad

Los selectores de atributo existen desde el CSS2. Con CSS3, las posibilidades son aún mayores.

Ejemplos de sintaxis:

- `[atributo^="valor"]`: para aplicar un estilo a un elemento cuyo `atributo` comience exactamente por dicho `valor`.
- `[atributo$="valor"]`: para aplicar un estilo a un elemento cuyo `atributo` termine exactamente con dicho `valor`.
- `[atributo*="valor"]`: para aplicar un estilo a un elemento cuyo `atributo` contenga al menos una vez ese `valor`.

2. Los vínculos hacia una dirección de e-mail

Vamos a ver en primer lugar un ejemplo de cómo aplicar un formato específico exclusivo para los vínculos que apunten hacia direcciones de correo electrónico.

Veamos qué estilos se han usado. Queremos seleccionar los vínculos que tengan el atributo`mailto` para aplicarles un estilo específico: un pequeño icono.

```
a {
   text-decoration: none;
}
a[href^="mailto"] {
   background: url(persona.png) left center no-repeat;
   padding-left: 30px;
}
```

Veamos ahora el código HTML de la página. Los dos primeros vínculos tienen el atributo `mailto`, el tercero no.

```
<h3>Los interlocutores a su servicio:</h3>
<p><a href="mailto: jserrano@dominio.com">Javier SERRANO VEGA:
Director comercial</a></p>
<p><a href="mailto: vespe@dominio.com">Victoria ESPERANZA CASTILLO:
Jefa de ventas</a></p>
<p>Para las reclamaciones, vaya a <a href="reclamaciones.html">
Problemas con el pedido</a></p>
```

Y el resultado obtenido:

3. Los vínculos de descarga

Veamos otro ejemplo de aplicación: en una página que permite descargar archivos, queremos diferenciar la apariencia visual de los vínculos en función del tipo, dependiendo de la extensión del archivo que podemos descargar (.pdf y .zip, en nuestro ejemplo).

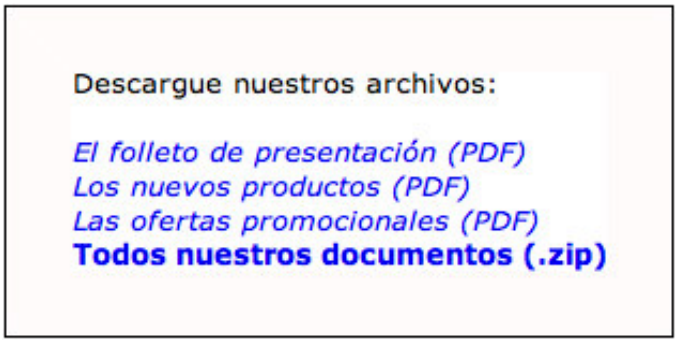
Ejemplos de selectores. Los vínculos de las extensiones .pdf (`a[href$=".pdf"]`) aparecerán en cursiva y los de las extensiones .zip (`a[href$=".zip"]`) en negrita.

```
a {
    text-decoration: none;
}
a[href$=".pdf"] {
    font-style: italic;
}
a[href$=".zip"] {
    font-weight: bold;
}
```

Veamos el código HTML de la página:

```
<p>Descargue nuestros archivos:</p>
<p><a href="folleto.pdf">El folleto de presentación (PDF)</a><br/>
<a href="novedades.pdf">Los nuevos productos (PDF)</a><br/>
<a href="ofertas.pdf">Las ofertas promocionales (PDF)</a></br/>
<a href="todo.zip">Todos nuestros documentos (.zip)</a></p>
```

Y el resultado obtenido:



La pseudo-clase de vínculo

1. Funcionalidad

Para crear vínculos dentro de una misma página, usaremos una sintaxis de este tipo:

```
<a href="#intro">Ir a la introducción</a>
```

Con la nueva pseudo-clase `:target`, usted podrá resaltar los vínculos que el visitante ya haya utilizado.

2. La aplicación

Veamos un ejemplo muy sencillo: usted tiene un documento muy largo, en el cual ha insertado un índice, al comienzo del mismo, con vínculos que apuntan hacia diferentes puntos situados en diversos lugares de la página.

Esta sería la estructura en HTML:

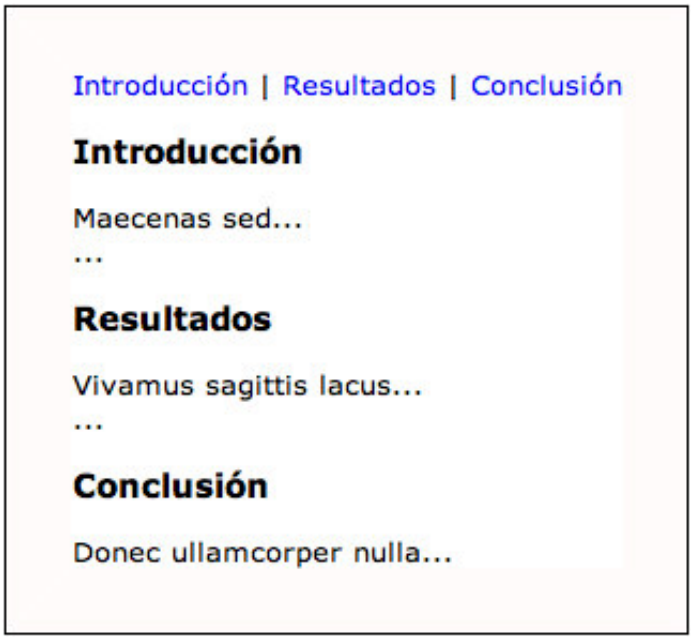
```
<p><a href="#intro">Introducción</a> | <a
href="#resultados">Resultados</a> | <a
href="#conclusion">Conclusión</a></p>
<h3><a id="intro">Introducción</a></h3>
<p>Maecenas sed...</p>
...
<h3><a id="resultados">Resultados</a></h3>
<p>Vivamus sagittis lacus...</p>
...
<h3><a id="conclusion">Conclusión</a></h3>
<p>Donec ullamcorper nulla...</p>
...
```

Y estos son los estilos CSS: cuando se use un vínculo del índice, se le aplicará un color de fondo dorado al elemento al que apunte.

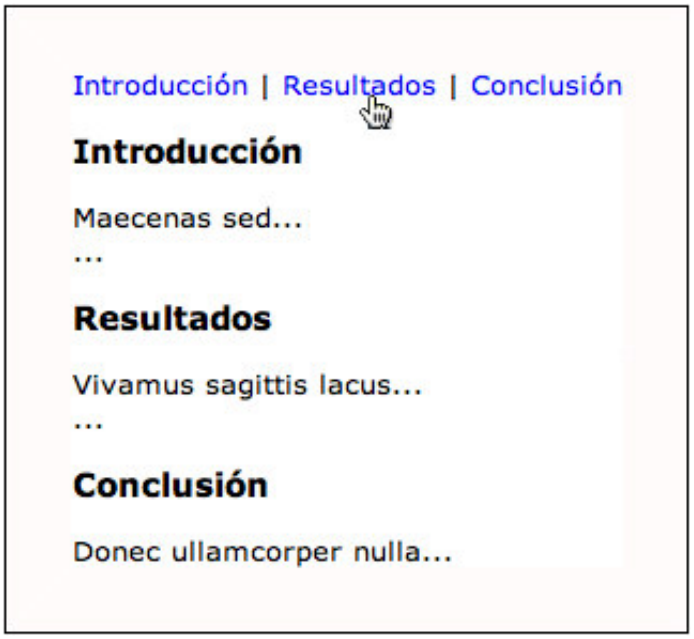
```
a {
     text-decoration: none;
}
a:target {
     background-color: gold;
}
```

3. Resultado

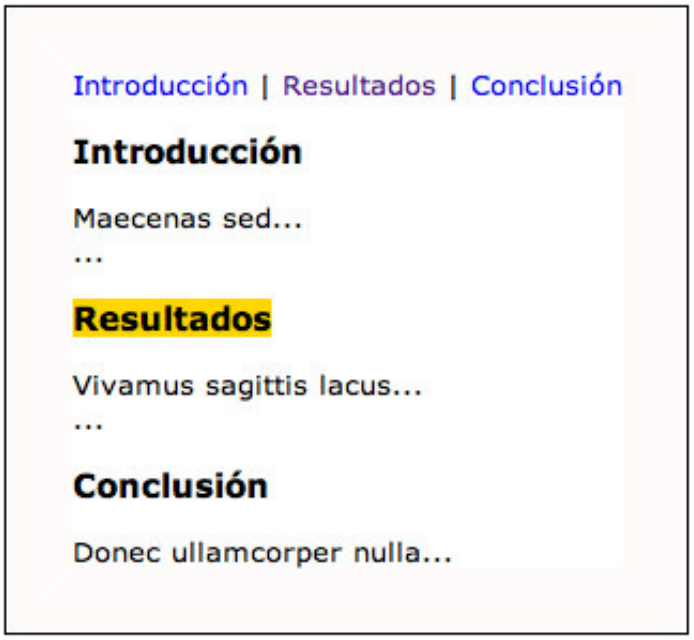
Así se visualizará la página cuando el usuario no haya realizado ninguna acción:



Supongamos ahora que el usuario hace clic en el índice, en el vínculo **Resultados**.



El usuario ha hecho clic en el vínculo **Resultados**, así que la zona vinculada de la página (`a:target`) aparece ahora resaltada en dorado (`background-color: gold`).



De este modo el usuario podrá saber en qué vínculos ya ha hecho clic.

Las pseudo-clases estructurales

1. Funcionalidad

El W3C nos propone toda una serie de pseudo-clases estructurales que nos permiten seleccionar con muchísima precisión los distintos elementos de la estructura de nuestras páginas web.

2. La raíz de la página

La pseudo-clase `root` permite seleccionar la raíz de la página web, es decir, el elemento`<html>`. La diferencia con el selector de elemento `html` es sencilla, en la cascada de estilos `:root`tiene prioridad.

3. El último elemento hijo

En CSS2 teníamos la pseudo-clase `:first-child` para seleccionar el primer elemento descendiente de un elemento padre. El CSS3 nos propone toda una retahíla de pseudo-clases que nos permiten seleccionar con precisión uno u otro elemento hijo.

La pseudo-clase `:last-child` permite seleccionar el último elemento hijo de un elemento padre. Este sigue el mismo principio de `:first-child`. Ambas pseudo-clases resultan en la actualidad muy prácticas a la hora de gestionar la fusión de los márgenes de párrafo en una caja`<div>`.

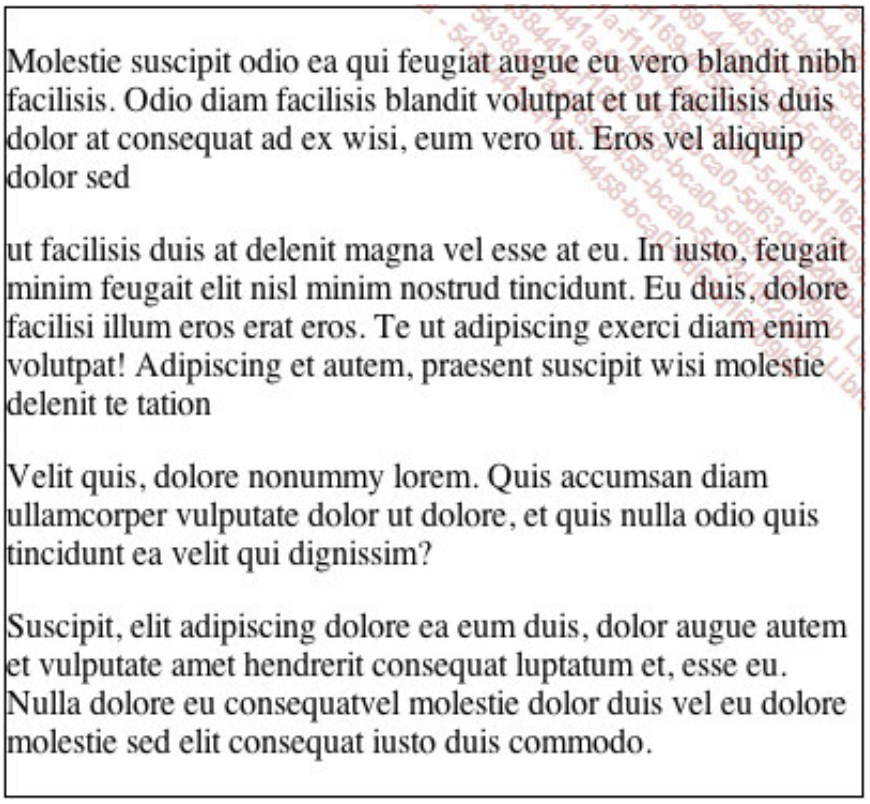
Esta es la estructura HTML de la página:

```
<div id="especial">
   <p>Molestie suscipit...</p>
   <p>ut facilisis duis at delenit...</p>
   <p>Velit quis, dolore...</p>
   <p>Suscipit, elit adipiscing dolore ea eum...</p>
</div>
```

Veamos el CSS para la caja `<div>`:

```
#especial {
   width: 400px;
   border: solid 1px black;
}
```

Y el resultado visual obtenido:

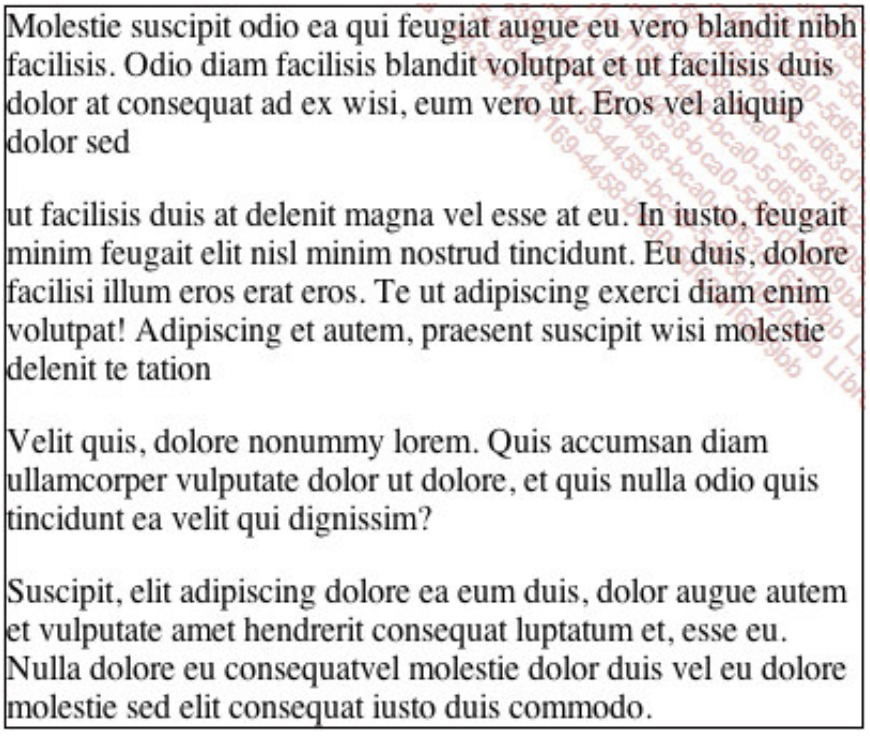


Como podemos ver, el margen superior del primer párrafo y el margen inferior del último párrafo aparecen en la caja `<div>`.

Para quitar esos márgenes, podemos usar el selector:

```
#especial p:first-child, #especial p:last-child {
   margin-top: 0;
   margin-bottom: 0;
}
```

Esta sería la visualización si usamos esas dos pseudo-clases:



Los márgenes superior e inferior son ahora de 0.

4. Los elementos descendientes

Para ver cómo se aplica esta pseudo-clase, vamos a tomar como ejemplo una lista `<ul>`.

Este sería el código HTML de una sencillísima lista:

```
<ul>
   <li>Rojo</li>
   <li>Verde</li>
   <li>Azul</li>
   <li>Gris</li>
   <li>Violeta</li>
   <li>Amarillo</li>
   <li>Negro</li>
</ul>
```

Puede hacer lo mismo con las líneas `<tr>` de una tabla.

El estilo CSS será siempre el mismo:

```
li:nth-child(x){
    background-color: gold;
}
```

La pseudo-clase `:nth-child(x)` permite seleccionar el enésimo elemento hijo de un elemento padre. Ese número `x` es el argumento de los paréntesis.

- El argumento `x` puede ser una cifra: `:nth-child(2)` para seleccionar al segundo hijo.

El resultado visual:

- El argumento `x` puede ser una palabra clave: `:nth-child(odd)` para seleccionar los hijos impares y `:nth-child(even)` para seleccionar los hijos pares. ¡Resulta ideal para crear alternancias!

El resultado visual:

- El argumento `x` puede ser un cálculo matemático de tipo: `an+b`. `n` para representar un valor que comience con `0` y que se incremente con valores distintos de `1`. El valor de `n` puede ser negativo o positivo.

Primer ejemplo:

`:nth-child(n+3)` permite saltarse los dos primeros hijos.

Este sería el cálculo:

`0+3=3`, para el tercer hijo,

`1+3=4`, para el cuarto hijo,

`2+3=5`, para el quinto hijo...

El resultado visual:

Segundo ejemplo:

`:nth-child(-n+3)` permite seleccionar los tres primeros hijos.

Este sería el cálculo:

`-0+3=3`, para el tercer hijo,

`-1+3=2`, para el segundo hijo,

`-2+3=1`, para el primer hijo,

`-3+3=0`, para ningún hijo,

`-4+3=-1`, para ningún hijo...

El resultado visual:

Tercer ejemplo:

`:nth-child(3n)` permite seleccionar un hijo de cada 3.

Este sería el cálculo:

`3x0=0`, para ningún hijo,

`3x1=3`, para el tercer hijo,

`3x2=6`, para el sexto hijo,

`3x3=9`, para el noveno hijo...

La visualización:

Cuarto ejemplo:

`:nth-child(3n+1)` permite seleccionar un hijo de cada 3 comenzando por el primer hijo.

Este sería el cálculo:

`(3x0)+1=1`, para el primer hijo,

`(3x1)+1=4`, para el cuarto hijo,

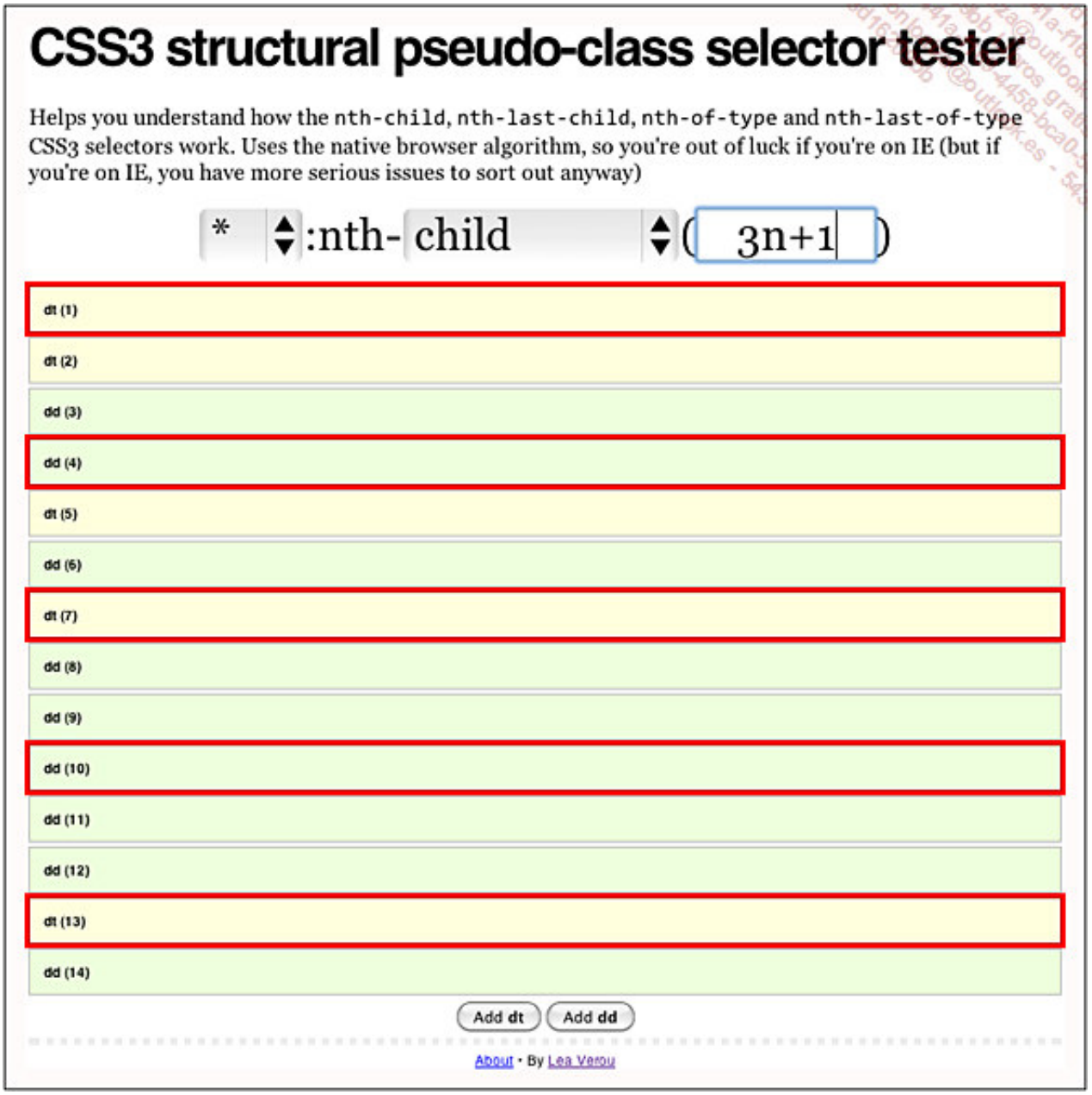
`(3x2)+1=7`, para el séptimo hijo,

`(3x3)+1=10`, para el décimo hijo.

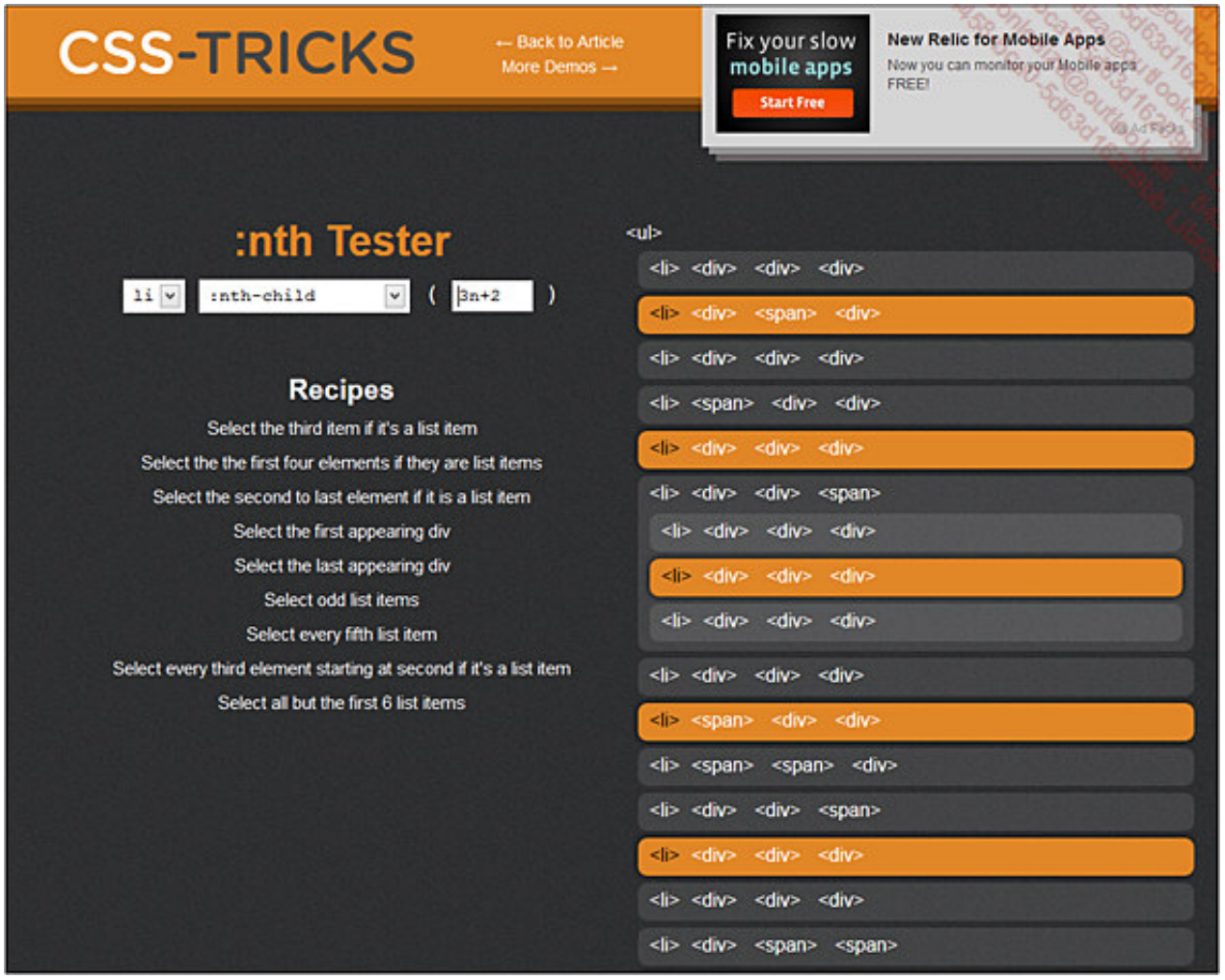
La visualización:

Le recomiendo dos herramientas en línea para probar la pseudo-clase `:nth-child(3n+1)`.

La página web **CSS3 structural pseudo-class selector tester** de Lea Verou (http://leaverou.me/demos/nth.html). Introduzca la fórmula que desee en el campo de texto y podrá ver en una tabla las líneas que resultarán seleccionadas en función de su fórmula.



Otra página web que sigue el mismo principio es **:nth-child tester** del sito web CSS-TRICKS (http://css-tricks.com/examples/nth-child-tester/).



5. Los últimos hijos de un elemento

La nueva pseudo-clase `:nth-last-child()` permite seleccionar los x últimos elementos hijos de un elemento padre. Su funcionamiento es similar al de `:nth-child()`.

6. Los primeros y los últimos elementos de un tipo determinado

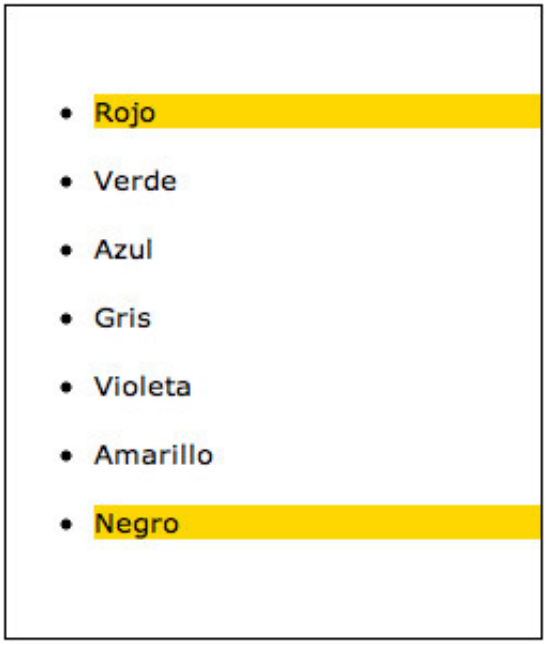
Las nuevas pseudo-clases `:first-of-type` y `:last-of-type` permiten seleccionar el primer y el último elemento de un tipo específico.

Si volvemos al ejemplo anterior de la lista `<ul>`, podremos seleccionar el primer y el último elemento`<li>`.

El estilo CSS sería:

```
li:first-of-type, li:last-of-type {
   background-color: gold;
}
```

El resultado visual:



Otro ejemplo con una tabla: vamos a seleccionar ahora todas las primeras celdas (`<td>`) de una tabla.

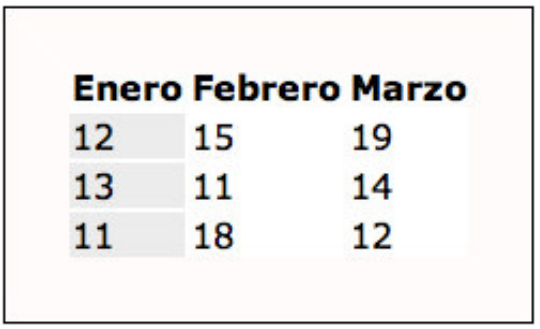
El código HTML de la tabla:

```
<table>
   <tr><th>Enero</th><th>Febrero</th><th>Marzo</th></tr>
   <tr><td>12</td><td>15</td><td>19</td></tr>
   <tr><td>13</td><td>11</td><td>14</td></tr>
   <tr><td>11</td><td>18</td><td>12</td></tr>
</table>
```

El estilo CSS:

```
td:first-of-type {
   background-color: #ececec;
}
```

El resultado visual:



7. Los primeros elementos de un tipo determinado

La pseudo-clase `:nth-of-type()` permite seleccionar el enésimo elemento de un tipo específico. Los argumentos posibles son los mismos que vimos con la pseudo-clase `:nth-child()`.

Veamos un ejemplo muy sencillo en el que se le ha aplicado un formato (`<strong>` y `<em>`) a algunas palabras de un párrafo (`<p>`):

```
<p>Donec ullamcorper <strong>nulla</strong> non metus auctor
<em>fringilla</em>. Fusce dapibus, tellus ac cursus commodo,
<strong>tortor</strong> mauris condimentum nibh, ut fermentum
massa justo sit amet risus. Cum sociis natoque
<strong>penatibus</strong> et magnis dis parturient montes.</p>
```

El elemento `<p>` presenta como primer elemento hijo el elemento `<strong>`, como segundo elemento hijo `<em>` y como tercer y cuarto elementos hijo nuevamente `<strong>`. Nosotros queremos seleccionar el segundo elemento `<strong>`, y no el segundo elemento hijo.

Veamos el estilo CSS:

```
strong:nth-of-type(2) {
   background-color: gold;
}
```

Y se mostraría así:

La clase `:nth-last-of-type()` permite seleccionar el enésimo último elemento de un tipo específico.

8. Los elementos únicos

La pseudo-clase `:only-child` permite seleccionar aquellos elementos que no tengan hermanos. Veamos por ejemplo un texto `<p>` en el cual hemos resaltado elementos con `<strong>`. Supongamos que queremos resaltar los párrafos que solamente contengan un único elemento`<strong>`.

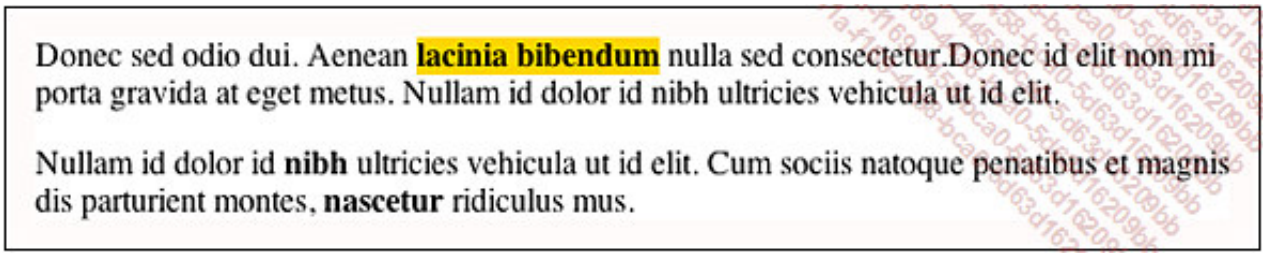
Este sería el código HTML. El segundo párrafo `<p>` contiene dos elementos `<strong>`, el primer párrafo solamente contiene uno.

```
<p>Donec sed odio dui. Aenean <strong>lacinia
bibendum</strong> nulla sed consectetur.Donec id elit non mi
porta gravida at eget metus. Nullam id dolor id nibh ultricies
vehicula ut id elit.</p>
<p>Nullam id dolor id <strong>nibh</strong> ultricies vehicula
ut id elit. Cum sociis natoque penatibus et magnis dis
montes, <strong>nascetur</strong> ridiculus mus.</p>
```

Veamos el estilo CSS utilizado:

```
strong:only-child {
   background-color: gold;
}
```

Y el resultado visual obtenido:



9. Los elementos hijo único de un tipo determinado

La pseudo-clase `:only-of-type` permite seleccionar los elementos de un tipo determinado que no tengan hermanos.

Volvamos al ejemplo anterior y añadamos un formato con `<em>` en el primer párrafo.

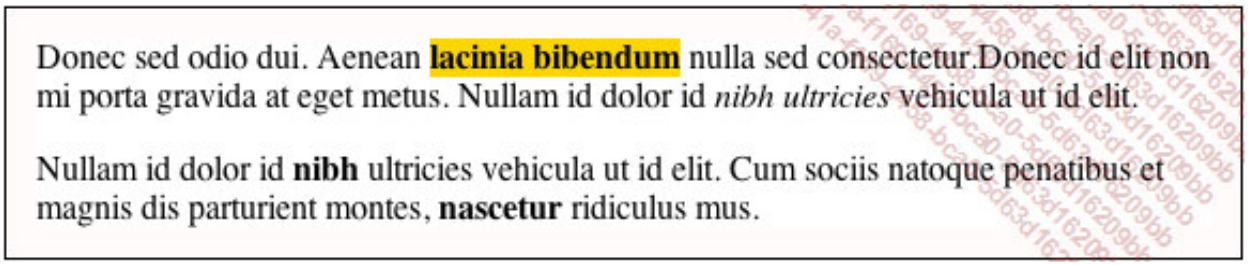
```
<p>Donec sed odio dui. Aenean <strong>lacinia
bibendum</strong> nulla sed consectetur.Donec id elit non mi
porta gravida at eget metus. Nullam id dolor id <em>nibh
ultricies</em> vehicula ut id elit.</p>
<p>Nullam id dolor id <strong>nibh</strong> ultricies vehicula
ut id elit. Cum sociis natoque penatibus et magnis dis parturient montes,
<strong>nascetur</strong> ridiculus mus.</p>
```

Con el estilo CSS anterior, el elemento `<strong>` del primer párrafo ya no aparecerá resaltado con un fondo de color, ya que ahora ese párrafo contiene dos elementos de formato. El elemento`<strong>` ya no está solo, ahora también tenemos una etiqueta `<em>`. Con la pseudo-clase `:only-of-type` podemos especificar el tipo de elemento aislado.

Veamos el estilo CSS:

```
strong:only-of-type {
   background-color: gold;
}
```

Y el resultado visual:



10. Los elementos vacíos

La pseudo-clase `:empty` permite seleccionar los elementos vacíos, es decir, aquellos que no contienen otros elementos. Esto resulta bastante práctico para las páginas web creadas dinámicamente. Podemos tener elementos (`<li>`, `<td>`, `<strong>`...) que estén vacíos. Con esta pseudo-clase podemos resaltarlos para localizarlos fácilmente.

La siguiente lista `<ul>` contienen dos `<li>` vacíos:

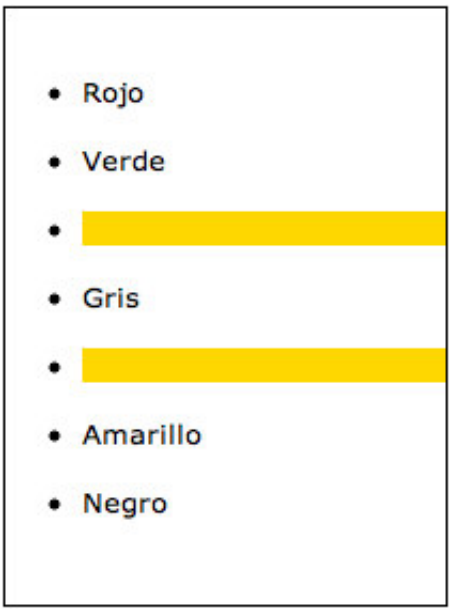
```
<ul>
   <li>Rojo</li>
   <li>Verde</li>
```

```
    <li></li>
    <li>Gris</li>
    <li></li>
    <li>Amarillo</li>
    <li>Negro</li>
</ul>
```

Veamos el estilo CSS utilizado:

```
li:empty {
    background-color: gold;
}
```

Y el resultado visual obtenido:



11. Todos los elementos excepto el elemento seleccionado

La pseudo-clase `:not()` es realmente muy práctica. Nos permite seleccionar todos los demás elementos a excepción del elemento especificado.

Tomemos como ejemplo un párrafo que contenga varios elementos de formato: `<strong>`, `<em>`, `<ul>`... Supongamos que en los párrafos `<p>` queremos cambiar la presentación visual de todos los elementos a excepción del elemento `<u>`.

El código HTML del párrafo:

```
<p>Et exerci feugait, <em>facilisis vulputate</em> feugiat,
hendrerit <u>veniam vero</u>? Ut praesent accumsan, nisl
consequatvel feugiat at duis ex <strong>vel ea dignissim</strong>
duis dignissim, velit qui, suscipit hendrerit vero. <em>Dolore
esse</em> luptatum blandit <u>eum commodo blandit</u> feugiat dolore
feugiat nisl veniam nibh veniam, et in nisl <strong>consequat
ut?</strong></p>
```

El estilo CSS:

```
p>:not(u){
    background-color: gold;
}
```

Y el resultado en pantalla:

Et exerci feugait, *facilisis vulputate* feugiat, hendrerit veniam vero? Ut praesent accumsan, nisl consequatvel feugiat at duis ex **vel ea dignissim** duis dignissim, velit qui, suscipit hendrerit vero. *Dolore esse* luptatum blandit eum commodo blandit feugiat dolore feugiat nisl veniam nibh veniam, et in nisl **consequat ut?**

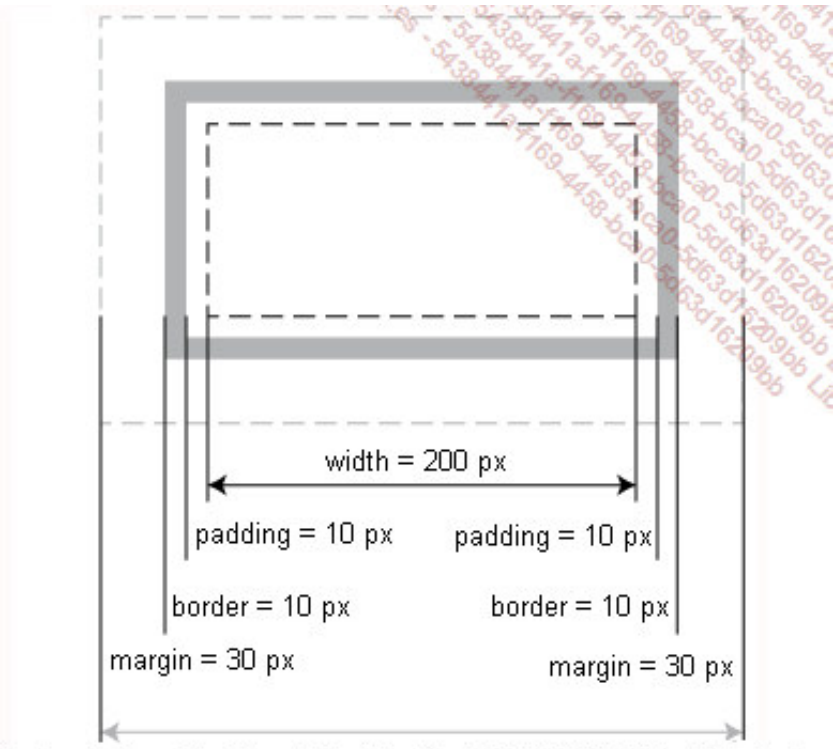
Las nuevas propiedades CSS3

De todas las novedades de CSS3, son las nuevas propiedades para el diseño de las cajas las que han generado un mayor entusiasmo y más reacciones de júbilo entre los diseñadores web. Todos soltaron un suspiro de alivio: ¡por fin vamos a poder aplicar bordes redondeados, degradados, sombras y transparencias a las cajas `<div>`!

El tamaño de las cajas

1. Con CSS 2.1

Como usted sabe, con CSS 2.1, el tamaño necesario para la visualización de una caja tiene en cuenta los márgenes (`margin`), los bordes (`border`), el espaciado (`padding`) y el contenido (`width`).



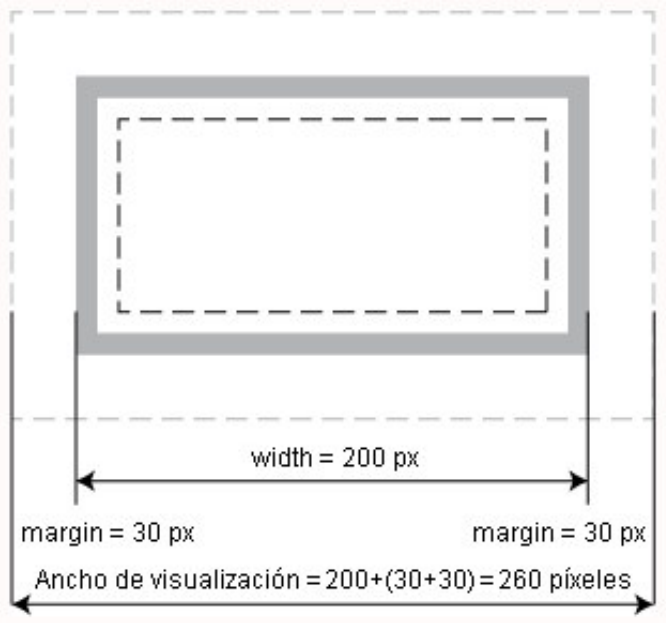
Ancho de visualización = 200+(10+10)+(10+10)+(30+30) = 300 píxeles

2. Con CSS3

En CSS3, con la propiedad `box-sizing` (http://www.w3.org/TR/css3-ui/#box-sizing), usted podrá indicar que los valores de `border` y de `padding` se incluyan en el tamaño de `width`.

Esta propiedad acepta dos valores:

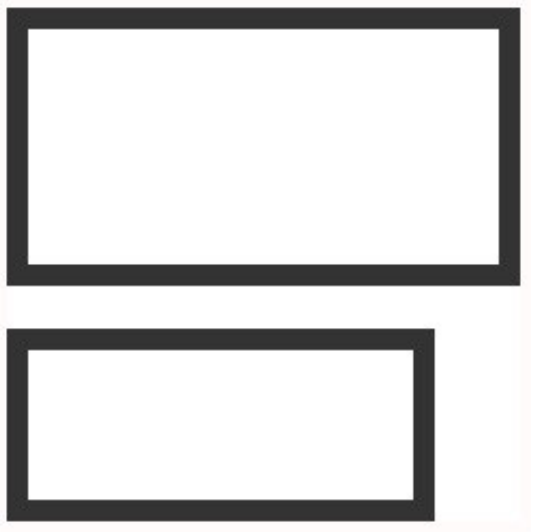
- `content-box`: el ancho de la caja se calcula en función del tamaño del contenido, del borde y del espaciado, como en CSS 2.1. Se trata del valor predeterminado.
- `border-box`: el ancho de la caja incluye el tamaño del contenido, del borde y del espaciado.



A continuación tenemos dos cajas con el mismo ancho de contenido (`width`), de espaciado (`padding`) y de borde (`border`). La caja `<div id="miCaja2">` presenta además la propiedad`box-sizing: border-box`.

```
#miCaja1 {
   position: absolute;
   top:  50px;
   left: 50px;
   width: 200px;
   height: 90px;
   padding: 10px;
   border: 10px solid #333;
   margin: 30px;
}
#miCaja2 {
   position: absolute;
   top: 200px;
   left: 50px;
   width: 200px;
   height: 90px;
   padding: 10px;
   border: 10px solid #333;
   margin: 30px;
   box-sizing: border-box;
}
```

Así se visualizan ambas cajas:



El desbordamiento de contenido

Los estilos CSS 2.1 nos permiten usar la propiedad `overflow` para gestionar el contenido cuando este último es mayor que el contenedor. Se trata del famoso principio del "desbordamiento de contenido".

CSS3 (http://dev.w3.org/csswg/css-overflow-3) nos propone dos nuevas propiedades para gestionar el desbordamiento:

- `overflow-x`: para gestionar el desbordamiento horizontal.
- `overflow-y`: para gestionar el desbordamiento vertical.

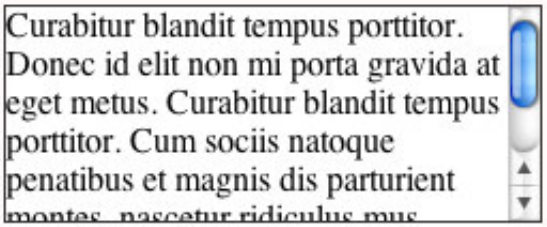
Los valores posibles son los mismos que en CSS 2.1:

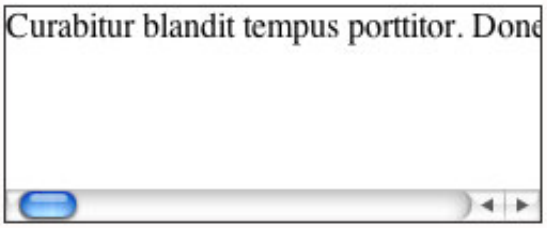
- `visible`: el contenido permanece siempre visible.
- `hidden`: el contenido permanece siempre oculto.
- `scroll`: se puede acceder al contenido gracias a una barra de desplazamiento que permanece siempre visible.
- `auto`: se puede acceder al contenido gracias a una barra de desplazamiento que solamente se muestra si resulta necesario.

Analicemos las dos cajas del siguiente ejemplo: la segunda, `#miCaja2`, usa las propiedades`overflow-x: scroll` y `white-space: nowrap`, que es indispensable para la visualización de la barra de desplazamiento.

```
#miCaja1 {
   width: 250px;
   height: 100px;
   border: 1px solid #333;
   padding: 0;
   overflow: auto;
   margin-bottom: 30px;
}
#miCaja2 {
   width: 250px;
   height: 100px;
   border: 1px solid #333;
   padding: 0;
   overflow-x: scroll;
   white-space: nowrap;
}
.sinMargen {
   margin: 0;
}
```

Veamos el resultado visual obtenido. La primera caja es `#miCaja1` y debajo tenemos la caja`#miCaja2`:


Curabitur blandit tempus porttitor. Donec id elit non mi porta gravida at eget metus. Curabitur blandit tempus porttitor. Cum sociis natoque penatibus et magnis dis parturient


Curabitur blandit tempus porttitor. Done

Los colores de fondo

1. El módulo de los colores

El W3C ha editado un módulo CSS3 específico para la gestión de los colores:http://www.w3.org/TR/css3-color/. Este módulo ya está finalizado, así que se presenta como**Recommendation** del 07 de junio de 2011.

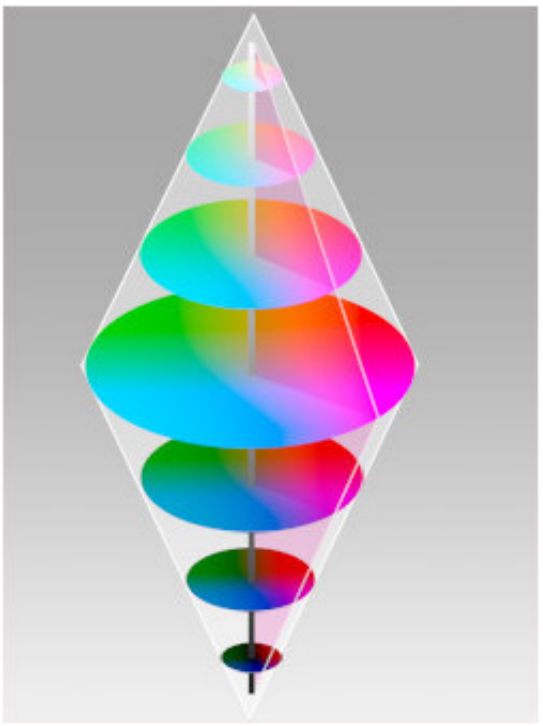
Las dos grandes novedades son el modo HSB y la transparencia.

2. El modo HSB

El modo **HSB** permite reunir en un solo y único gráfico (que se conoce como círculo cromático) los modelos colorimétricos **RGB** y **CMJN**, que en inglés se conoce como modo **HSL**.

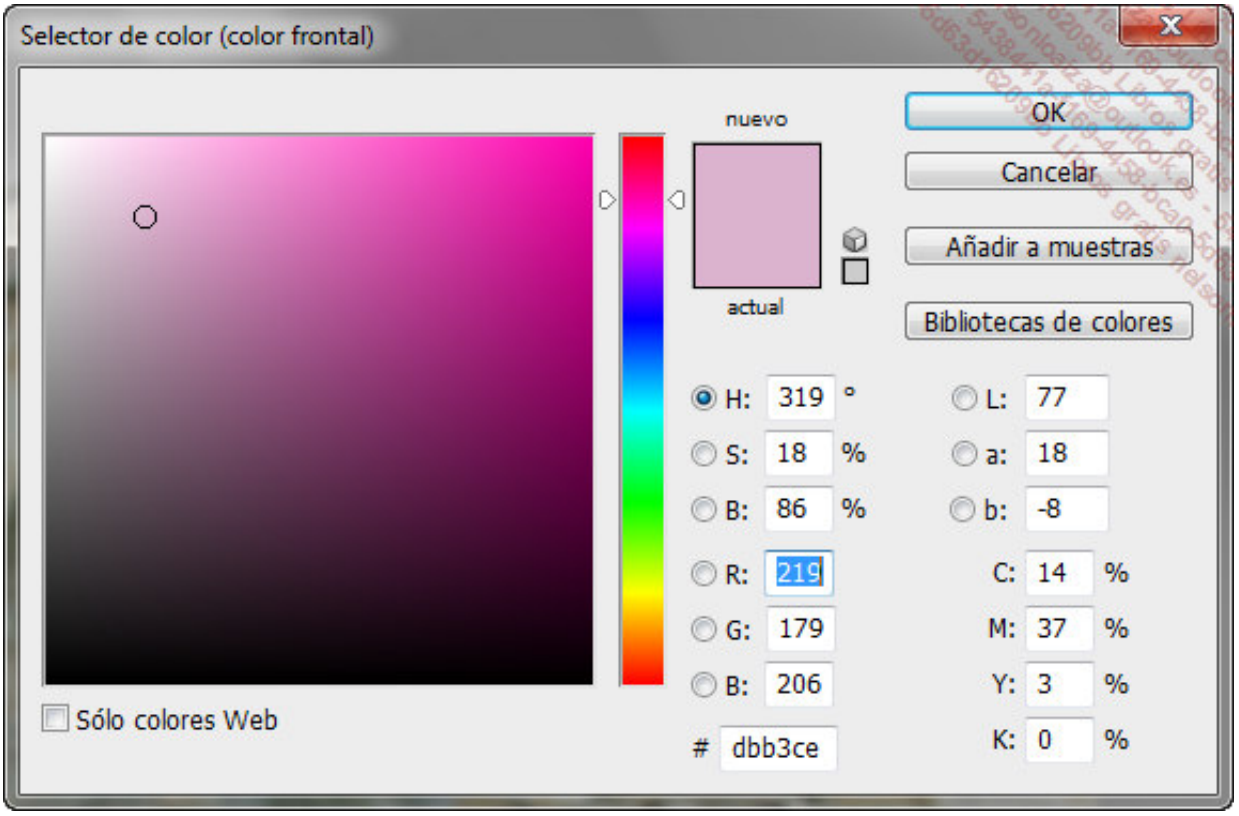
- La **H** corresponde a Tono (**Hue** en inglés): se trata del color. La unidad de medida es el grado. Por definición el rojo es 0º.
- La **S** corresponde a **Saturación**: se trata de la "pureza" del tono, es decir, de la cantidad de gris presente en un tono. Cuanto más saturado esté un color, este será más intenso, más vivo, ya que contendrá muy poco gris. Cuanto menos saturado esté un color, este será más apagado, sin brillo, ya que contendrá mucho gris. La unidad es el %. Un tono al 0% será totalmente gris (sin saturación), un tono al 100% será totalmente "puro" (saturado), sin una pizca de gris.
- La **B** corresponde a **Brillo** (**Brightness** en inglés): se trata de la cantidad de luz. La unidad varía de 0% de luz, el negro, a 100% de luz, el blanco.

Así se representa gráficamente el modo HSB:



(Fuente: http://diagexpertonline.com/ESP/index-3.html)

Este es el selector de color en modo HSB de Adobe Photoshop:



En el ejemplo, el color rosa seleccionado presenta el valor HSB: `hsl(319,18%,86%)`.

Veamos el estilo CSS, con dicho color HSB:

```
#rosa {
   width: 100px;
   height: 50px;
   background-color: hsl(319,18%,86%);
}
```

3. El modo HSB con transparencia

Con la unidad `hsla`, es posible añadir un cuarto valor opcional, la transparencia. Su valor va de `0`, totalmente transparente, a `1`, totalmente opaco (sin transparencia).

En el ejemplo siguiente tenemos dos cajas `<div>` superpuestas y la que está por encima utiliza la transparencia.

Los estilos CSS:

```
#verde {
   position: absolute;
   top: 10px;
   left: 10px;
   width: 100px;
   height: 50px;
   background-color: hsl(153,39%,74%);
}
#amarillo {
   position: absolute;
   top: 30px;
   left: 50px;
   width: 100px;
   height: 50px;
   background-color: hsla(40,50%,80%,.5);
```

```
}
```

Y el resultado visual:

4. El modo RGB con transparencia

Al igual que para el TSL con transparencia, el modo **RGB** (del inglés *Red, Green, Blue*, es decir, Rojo, Verde, Azul), ahora acepta la transparencia en unidades `rgba()`.

Ejemplo: `background-color: rgba(200,20,10,.5);`

5. Un editor RGBA en línea

El sitio web **CSS 3.0 Maker** (http://www.css3maker.com/index.html) pone a su disposición multitud de herramientas en línea para generar estilos CSS3. La herramienta RGBA (http://www.css3maker.com/css-3-rgba.html) le permite seleccionar un color con la unidad `rgba()`.

En esa interfaz oscura (y poco práctica), en el recuadro **CSS3 Styles**, arrastre los selectores para escoger los componentes rojo, verde y azul que desee. En el recuadro **CSS3 Preview Area**, podrá visualizar el color que haya seleccionado. En el recuadro **CSS3 Codeview**, podrá copiar el código generado, o bien, puede hacer clic en el botón **Download**.

6. La transparencia

La nueva propiedad `opacity` permite aplicar transparencias a todos los elementos HTML y a sus descendientes. Conviene que prestemos mucha atención a la hora de aplicarla. Al igual que antes, los valores van de 0, para la transparencia, a 1, para la opacidad.

Veamos un ejemplo en el que se ha aplicado la transparencia a una caja `<div>`:

```
#transp {
   width: 250px;
   padding: 0;
   background-color: lightgreen;
   opacity: .3;
   text-align: center;
}
```

Este sería el código HTML:

```
<div id="transp">
   <h1>Título de nivel 1</h1>
   <p>Magna Nibh Malesuada</p>
</div>
```

Y se visualizará así. La caja (`<div>`) y su contenido (`<h1>` y `<p>`) tienen una transparencia de `0.3`.

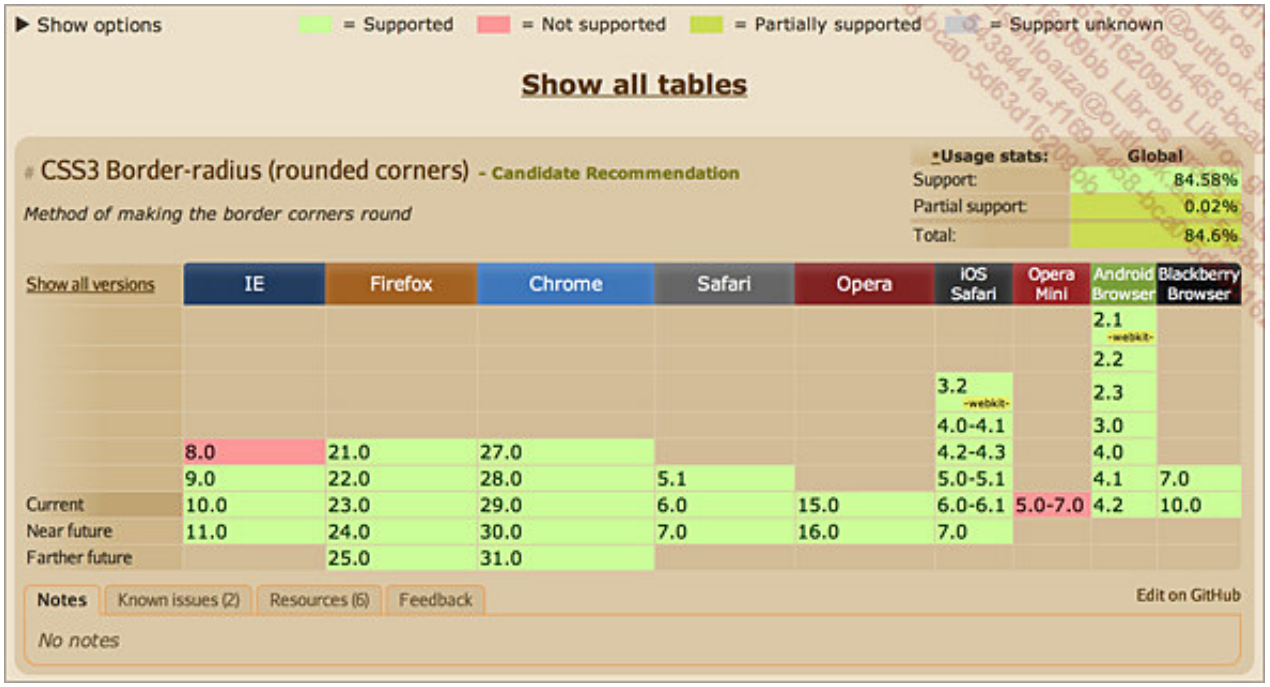
Situación actual de fondos y bordes

Este módulo hace referencia a los fondos, los bordes, las esquinas redondeadas y las sombras. A nivel del W3C, este módulo está casi finalizado, a 24 de julio de 2012 se encuentra en la fase de**Candidate Recommendation**: http://www.w3.org/TR/css3-background/

Los bordes redondeados

1. Los prefijos

En la actualidad (septiembre 2013), todos los navegadores modernos reconocen las propiedades CSS3 de bordes redondeados (http://www.w3.org/TR/css3-background/#corners), por tanto ya no es necesario añadir los prefijos de los navegadores.



2. Bordes idénticos

Los "contenedores" (`<div>`, `<section>`, `<header>`...) ya admiten los bordes redondeados. El valor de la curva expresado en píxeles puede ser idéntico para las cuatro esquinas, o bien, diferente para cada una de ellas. Este valor indica el radio del "círculo" que determina el ángulo de cada contenedor o de cada caja.

Este es el estilo CSS de una caja `<div>` con los cuatro bordes idénticos, con una curva de 10 píxeles:

```
#esquinas-iguales{
   border-radius: 10px;
   position: absolute;
   top: 10px;
   left: 10px;
   width: 200px;
   height: 90px;
   background-color: lightblue;
}
```

Fíjese en que Opera e IE reconocen de forma nativa la propiedad `border-radius`.

Y el resultado visual:

3. Bordes redondeados diferentes

Veamos ahora una caja con bordes redondeados diferentes para cada ángulo:

```
#esquinas-diferentes {
   border-radius: 10px 30px 20px 40px;
   position: absolute;
   top: 130px;
   left: 10px;
   width: 200px;
   height: 90px;
   background-color: lightblue;
}
```

Como puede ver, se han usado dos sintaxis:

Una sintaxis para cada ángulo:

```
border-top-left-radius: 10px;
border-top-right-radius: 30px;
border-bottom-right-radius: 20px;
border-bottom-left-radius: 40px;
```

Y la sintaxis corta, con este orden: el ángulo superior izquierdo, el superior derecho, el inferior derecho y el inferior izquierdo.


```
border-radius: 10px 30px 20px 40px;
```

Y el resultado visual:

4. Elipses en los bordes

Si lo desea, puede aplicar elipses, en lugar de círculos, a los bordes de sus cajas.

Bastará con que indique dos valores separados mediante una barra oblicua /. El primer valor corresponderá al radio horizontal y el segundo al radio vertical.

Veamos una caja con los bordes redondeados en elipse. El radio horizontal es de 40 píxeles y el radio vertical de 20 píxeles.

```
#esquinas-en-elipse {
   border-radius: 40px/20px;
   position: absolute;     top: 300px;
   left: 10px;     width: 200px;
   height: 90px;
   background-color: lightblue;
}
```

Y el resultado obtenido:

También puede crear, por qué no, una caja con cuatro esquinas redondeadas diferentes en elipse. La primera serie de valores indicará el radio horizontal, en este orden: el ángulo superior izquierdo, el superior derecho, el inferior derecho y el inferior izquierdo. La segunda serie indicará el radio vertical, en el mismo orden.

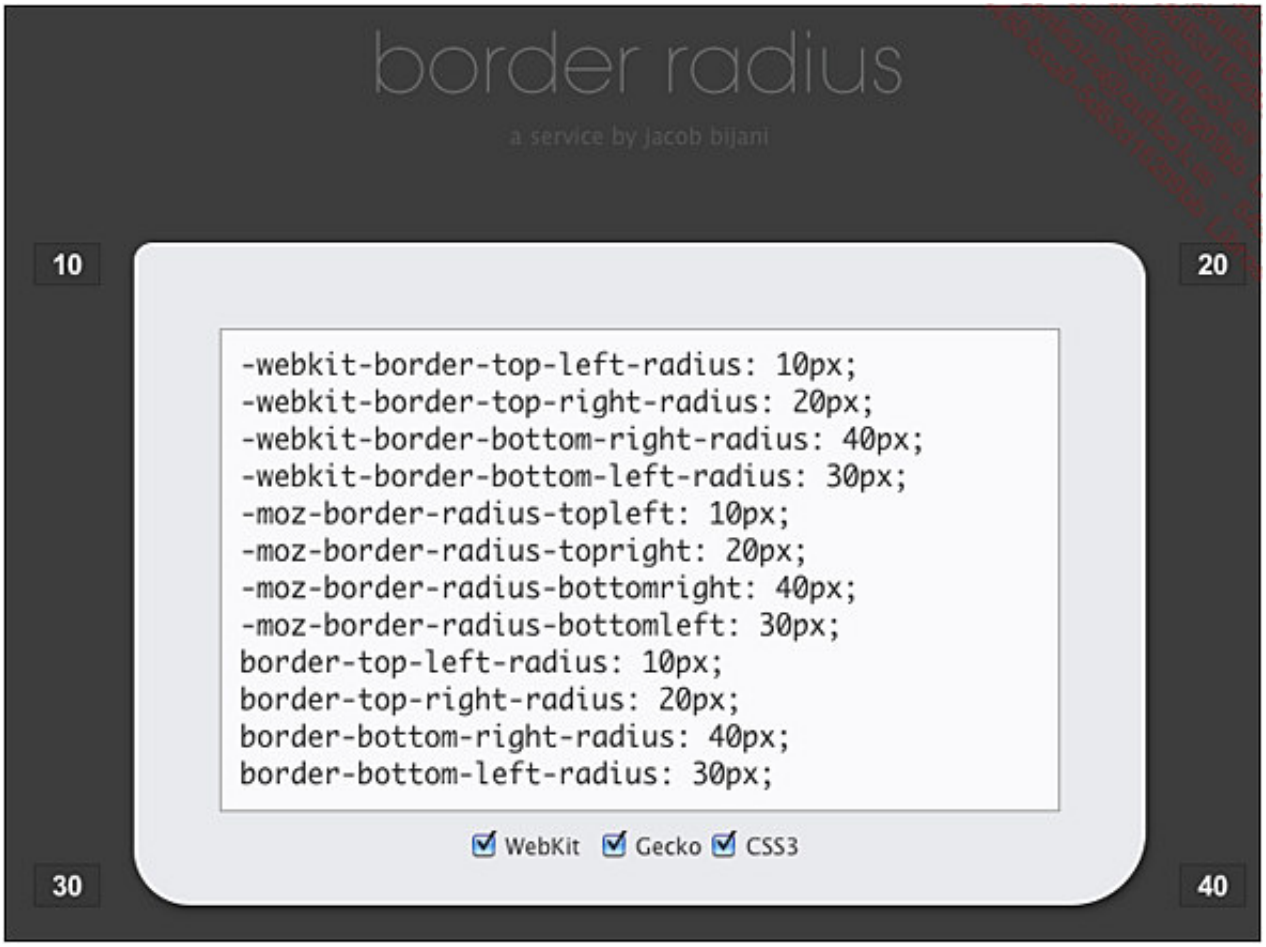
```
#esquinas-en-elipse-2 {
   border-radius: 40px 30px 50px 10px / 20px 10px 10px 40px;
   position: absolute;
   top: 450px;
   left: 10px;
   width: 200px;
   height: 90px;
   background-color: lightblue;
}
```

El resultado visual de este ejemplo:

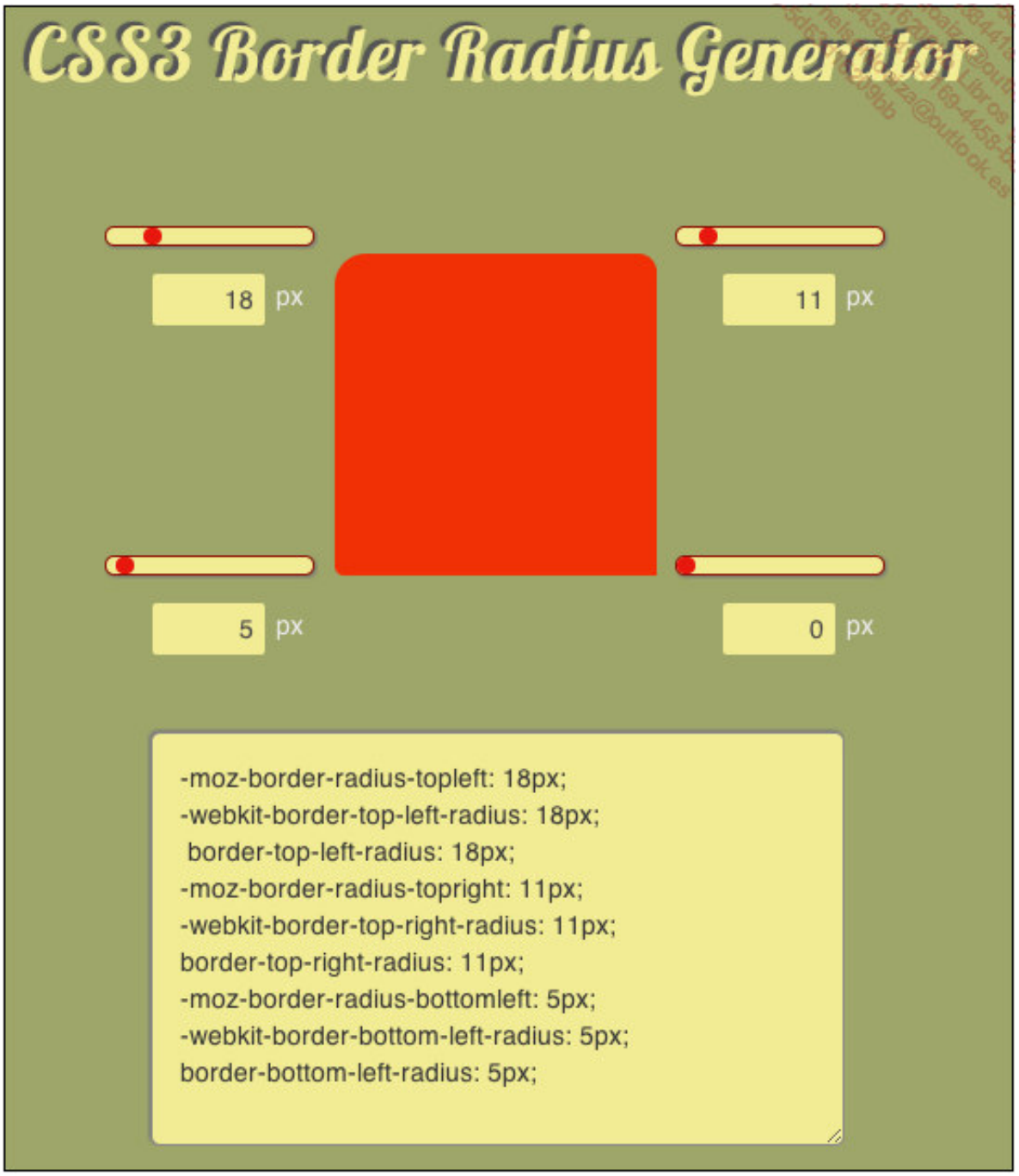
5. Los generadores en línea

Existen multitud de generadores CSS3 en línea. Veamos tres de ellos con los que podrá crear rápidamente un estilo CSS3.

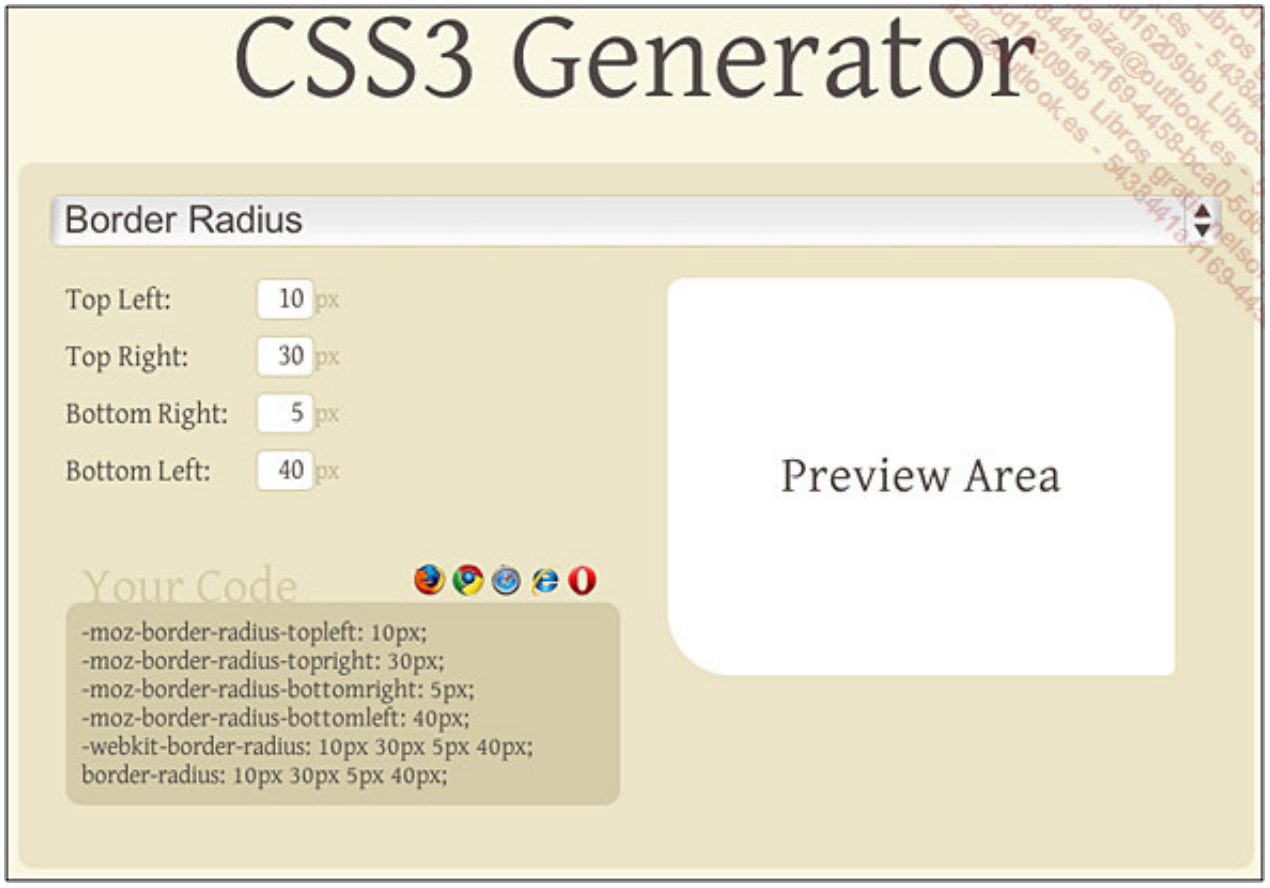
El sitio web **Border-Radius** (http://border-radius.com/) propone una interfaz minimalista, en la que deberá introducir los valores deseados para cada esquina. El ejemplo se mostrará inmediatamente. Seleccione los prefijos de los navegadores que desee, todo se hace automáticamente. Usted solamente tendrá que copiar y pegar el código.



El sitio web **CSS3GEN** (http://css3gen.com/) permite generar un gran número de propiedades CSS3. En cuanto a los bordes redondeados (http://css3gen.com/border-radius/), dispone de una interfaz en la que puede introducir los valores, o bien, puede utilizar un cursor. Nuevamente, el código se generará automáticamente.



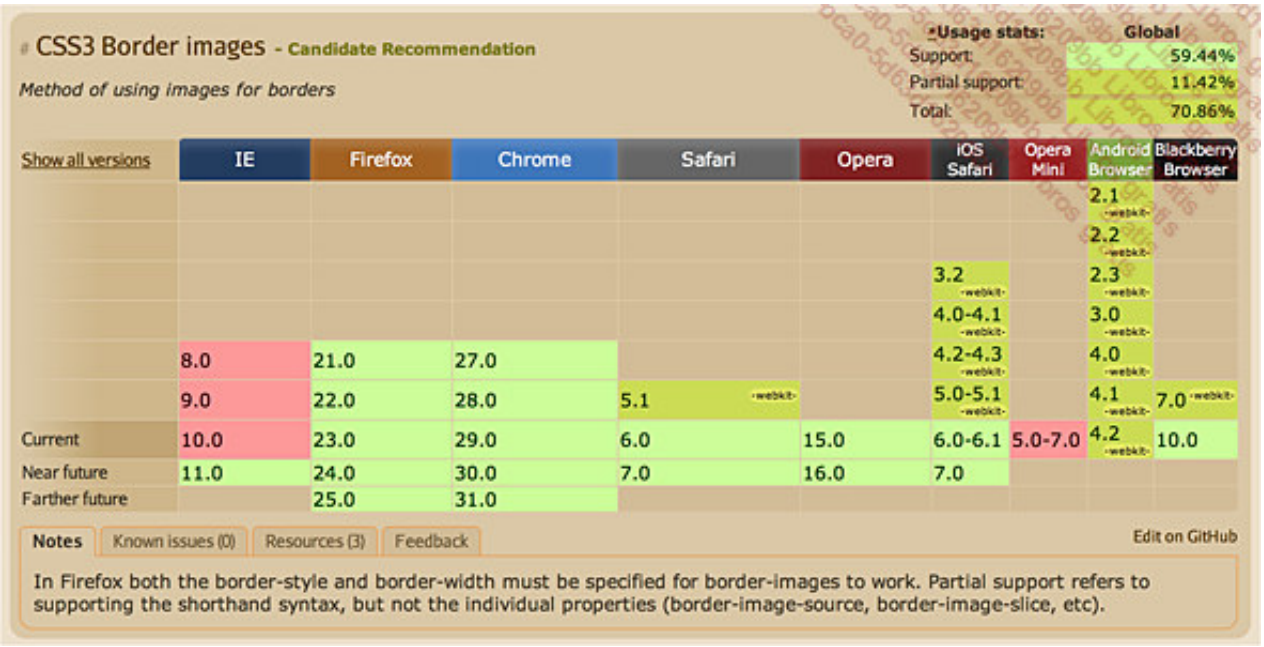
El sitio web **CSS3 Generator** (http://css3generator.com/) también propone diferentes propiedades que se pueden generar en línea. En el menú desplegable **Choose Something**, seleccione **Border Radius**. Especifique a continuación si desea aplicar valores idénticos, o no, a todos los ángulos. Se generará inmediatamente la visualización y el código, de modo que podrá copiarlo y pegarlo donde desee.


CSS3 Generator
Border Radius
Top Left: 10 px
Top Right: 30 px
Bottom Right: 5 px
Bottom Left: 40 px
Preview Area
Your Code
-moz-border-radius-topleft: 10px;
-moz-border-radius-topright: 30px;
-moz-border-radius-bottomright: 5px;
-moz-border-radius-bottomleft: 40px;
-webkit-border-radius: 10px 30px 5px 40px;
border-radius: 10px 30px 5px 40px;

Los bordes imaginativos

1. La sintaxis del W3C

Los estilos CSS3 para los bordes (http://www.w3.org/TR/css3-background/#border-images) ahora permiten aplicar imágenes a los bordes. Esa imagen se repetirá en el borde en función del patrón que haya definido el diseñador web. Cuidado, una vez más tendremos que usar los prefijos propietarios.



Comience creando la imagen que se repetirá en el borde, según un patrón que permita definir los cuatro ángulos y los cuatro lados.

En el ejemplo se trata de una imagen de 60 píxeles de lado. Tenemos un círculo para cada ángulo y un triángulo con una orientación diferente para cada uno de los cuatro lados. Cada "elemento decorativo" mide 20 píxeles de lado.

Utilice la propiedad `border-image`.

Esta es la sintaxis corta con los prefijos de navegación:

```
#borde {
   position: absolute;
   top: 10px;
   left: 10px;
   width: 300px;
   height: 100px;
   border-width: 20px;
   border-image: url(elemento.png) 20 round;
   background-color: lightyellow;
}
```

Como puede ver, se ha usado la propiedad estándar `border-width`, que en este ejemplo es igual a la dimensión de los elementos que se colocarán en el borde: `20px`.

Los argumentos de la propiedad `border-image` (con la sintaxis corta):

- `url(triangulo.png)` : ruta de acceso a la imagen.
- `20`: porción en píxeles de la imagen que se usará para cada elemento decorativo. Vamos a usar 20 píxeles de la imagen para cada elemento. Si el valor es el mismo para todos los lados, bastará con indicar un único valor, que será el equivalente de: `20 20 20 20`. Si los valores son diferentes para los distintos lados, indique dichos valores en este orden: arriba, derecha, abajo, izquierda.
- `round`: permite repetir la imagen y redimensionarla para que ocupe la totalidad del borde, sin que aparezca cortada (los dos otros valores son: `stretch` para estirar el motivo sobre toda la dimensión del lado y `repeat` para repetir el motivo, que podría aparecer cortado).

Y el resultado visual:

Aquí hemos usado la sintaxis corta: `border-image`. Usted podrá individualizar cada propiedad:

- `border-image-source`: URL de la fuente de la imagen.
- `border-image-slice`: los valores de corte de la imagen para obtener los motivos.
- `border-image-width`: grosor del borde.
- `border-image-outset`: distancia del borde de la imagen.
- `border-image-repeat`: tipo de repetición de la imagen.

2. Los generadores en línea

El sitio web **Border Image Generator** (http://border-image.com/) le permite generar el código CSS3 para crear bordes imaginativos.

En el campo **Image**, introduzca la URL de su imagen.

Use los cursores **Offset** para indicar el tamaño de los motivos.

También puede seleccionar tamaños diferentes para el ancho del borde en la zona **Border Size**.

En el menú desplegable **Border Repeat**, seleccione el estilo de repetición que desee.

Podrá visualizar el resultado en la zona **Preview**, así como el código generado.


border-image-generator
[toggle style]
Editor
Image:
Top Offset:
Right Offset:
Bottom Offset:
Left Offset:
Border Size
Border Repeat
Vertical Repeat:
Round
Horizontal Repeat:
Round
Preview
border-width: 20px;
-moz-border-image: url(http://kryztho.free.fr/motif.png) 20 round;
-webkit-border-image: url(http://kryztho.free.fr/motif.png) 20 round;
-o-border-image: url(http://kryztho.free.fr/motif.png) 20 round;
border-image: url(http://kryztho.free.fr/motif.png) 20 round;
incaseofstairs | @kpdecker | gradient-scanner | border-image-generator @ github

Las sombras

1. Situación actual

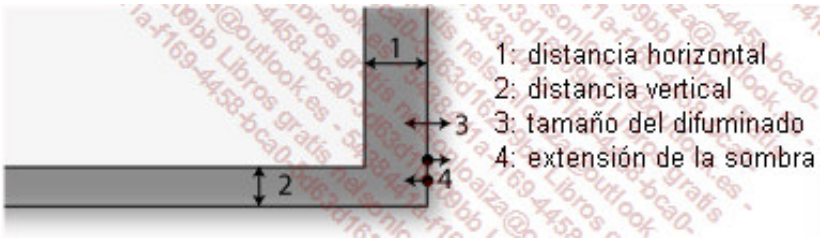
El módulo del CSS3 para el fondo y los bordes (http://www.w3.org/TR/css3-background) se encuentra, a 24 de julio de 2012, en fase de **Candidate Recommendation**, así que ya está casi terminado.

Actualmente, todos los navegadores reconocen la propiedad CSS3 `box-shadow`.

2. Crear una sombra

Se trata de la propiedad `box-shadow` la que permite aplicar una sombra a un elemento, a una caja`<div>`. Los atributos son:

- la distancia horizontal de la sombra.
- la distancia vertical de la sombra.
- el valor de difuminado de la sombra. Este atribulo es facultativo, si no lo especifica se usará el valor 0.
- el valor de extensión de la sombra. Este valor permite situar el "punto medio" del degradado, es decir, el punto a partir del cual la sombra comienza a desvanecerse. Un valor positivo alargará la sombra, un valor negativo la comprimirá. Este atribulo es facultativo, si no lo especifica se usará el valor 0.
- el color de la sombra.
- la posición de la sombra (`inset`, hacia el interior, o `outset`, hacia el exterior, que es el valor predeterminado).



Veamos un ejemplo sencillo:

```
#miCaja{
 position: absolute;
 top: 10px;
 left: 10px;
 width: 300px;
 height: 100px;
 background-color: lightyellow;
 box-shadow: 10px 15px 8px 2px rgb(12,34,56);
}
```

Esta sombra presenta un desplazamiento horizontal de 10 píxeles, un desplazamiento vertical de 15 píxeles, un difuminado de 8 píxeles, una extensión de 2 píxeles y es de color gris oscuro.

Y el resultado visual:

3. Varios ejemplos de sombras

La primera sombra no presenta ni difuminado, ni extensión.

```
#miCaja4 {
   ...
   box-shadow: 10px 15px 0 0 grey;
}
```

En este ejemplo podría haber omitido perfectamente los parámetros de difuminado y de extensión. He preferido dejarlos para visualizar mejor todos los parámetros.

En el siguiente ejemplo hemos desplazado y difuminado la sombra, pero no se le ha aplicado ninguna extensión:

```
#miCaja4 {
   ...
   box-shadow: 10px 10px 5px 0 grey;
}
```

Apliquemos ahora los mismos parámetros, pero con una extensión positiva: la sombra se agrandará en ese número de píxeles.

```
#miCaja4 {
   ...
   box-shadow: 10px 10px 5px 5px grey;
}
```

Y si aplicamos los mismos parámetros, pero con una extensión negativa, la sombra se reducirá de ese número de píxeles.

```
#miCaja4 {
   ...
   box-shadow: 10px 10px 5px -5px grey;
}
```

Un último ejemplo, sin desplazamiento, solamente con difuminado. De este modo la sombra aparecerá en los cuatro lados de la caja.

```
#miCaja4 {
   ...
   box-shadow: 0 0 20px 0 grey;
}
```

4. Aplicar varias sombras

Podemos aplicar varias sombras a un mismo elemento, separándolas mediante una coma. Tenga cuidado, el orden indicado es importante. La primera sombra indicada se colocará en primer lugar, y así sucesivamente. Comience por la sombra que se encuentre a menor distancia, de lo contrario quedará oculta detrás de las demás sombras que presenten una distancia mayor.

```
#miCaja2 {
   position: absolute;
   top: 10px;
   left: 10px;
   width: 300px;
   height: 100px;
   background-color: lightyellow;
   box-shadow: 5px 5px 3px 2px #ccc,
   10px 15px 8px 2px rgb(12,34,56);
}
```

Y el resultado visual:

Veamos una última caja con dos sombras desplazadas, una de las cuales hacia el interior de la caja (`inset`):

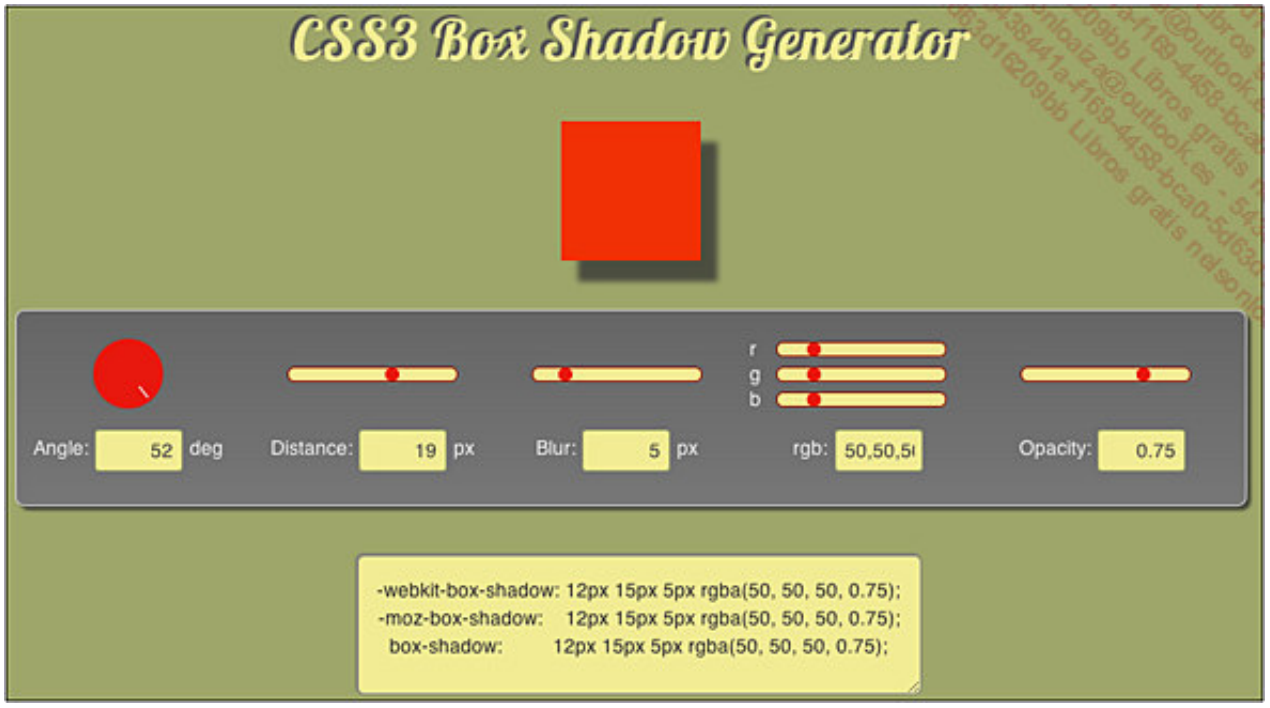
```
#miCaja2 {
   position: absolute;
   top: 150px;
   left: 10px;
   width: 300px;
   height: 100px;
   background-color: lightyellow;
   box-shadow: 5px 3px 2px #ccc inset,
   10px 15px 8px 2px rgb(12,34,56);
}
```

Y el resultado visual:

5. Los generadores en línea

Existen multitud de generadores de sombras en línea.

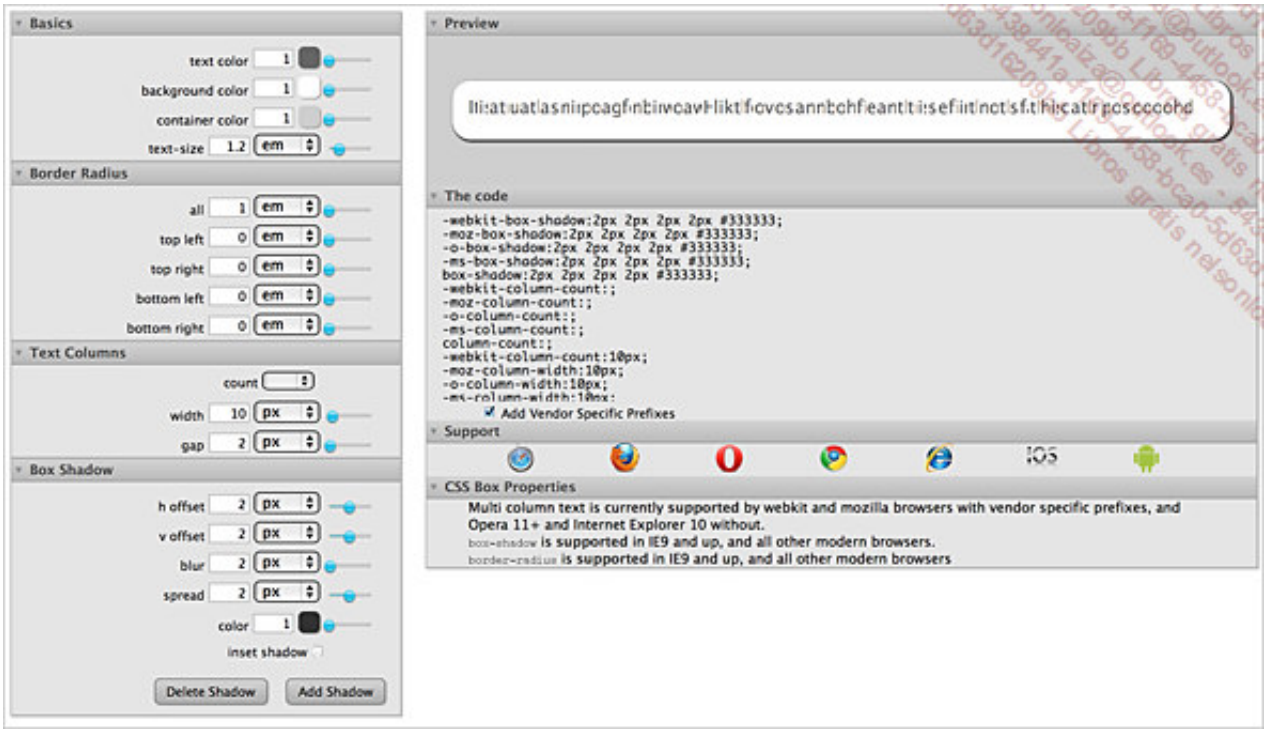
CSS3Gen (http://css3gen.com/box-shadow/) pone a su disposición una interfaz simple en la que podrá usar un ángulo para indicar el desplazamiento horizontal y vertical.



CSS3 Generator (http://css3generator.com/) también presenta una interfaz simple, pero un poco más completa.

El sitio web **WestCIV** propone un generador muy completo para trabajar con las distintas propiedades de las cajas (http://westciv.com/tools/box-properties/index.html). La ventaja de esta interfaz es que le permite insertar varias sombras.

En la zona **Box Shadow**, el botón **Add Shadow** le permite añadir las sombras. En el ejemplo, he creado tres sombras (cuidado, la captura de pantalla es un montaje, porque la ventana original era demasiado grande).

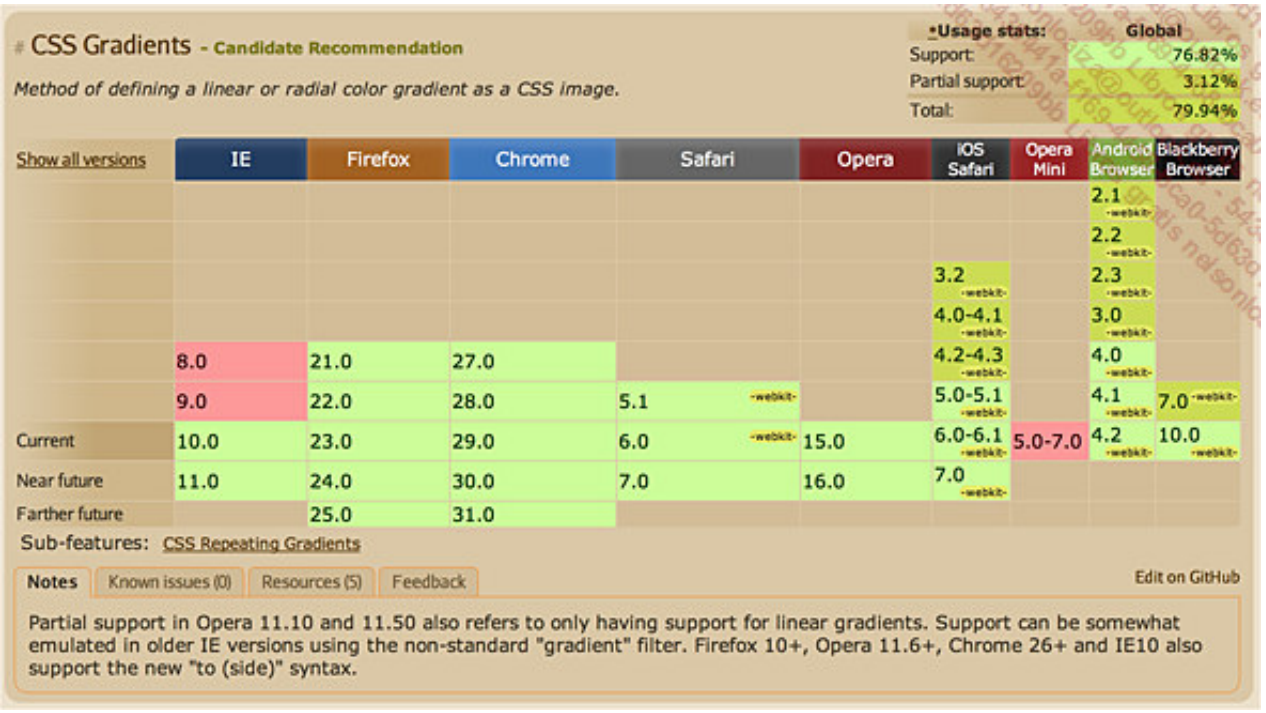

Basics
text color 1
background color 1
container color 1
text-size 1.2 em
Border Radius
all 1 em
top left 0 em
top right 0 em
bottom left 0 em
bottom right 0 em
Text Columns
count
width 10 px
gap 2 px
Box Shadow
h offset 2 px
v offset 2 px
blur 2 px
spread 2 px
color 1
inset shadow
Delete Shadow
Add Shadow
Preview
The code
-webkit-box-shadow:2px 2px 2px 2px #333333;
-moz-box-shadow:2px 2px 2px 2px #333333;
-o-box-shadow:2px 2px 2px 2px #333333;
-ms-box-shadow:2px 2px 2px 2px #333333;
box-shadow:2px 2px 2px 2px #333333;
-webkit-column-count:;
-moz-column-count:;
-o-column-count:;
-ms-column-count:;
column-count:;
-webkit-column-count:10px;
-moz-column-width:10px;
-o-column-width:10px;
Add Vendor Specific Prefixes
Support
iOS
CSS Box Properties
Multi column text is currently supported by webkit and mozilla browsers with vendor specific prefixes, and Opera 11+ and Internet Explorer 10 without.
box-shadow is supported in IE9 and up, and all other modern browsers.
border-radius is supported in IE9 and up, and all other modern browsers

Los degradados

1. Situación actual

El módulo CSS3 para las imágenes y los degradados se encuentra todavía en fase de **Candidate Recommendation** a 17 de abril de 2012: http://www.w3.org/TR/css3-images/

El reconocimiento de los degradados por parte de los navegadores es bastante bueno, pero resulta conveniente utilizar prefijos en algunos de ellos, sobre todo en Safari 6.



Observe que la tabla muestra que Firefox 23 reconoce los degradados, pero tras llevar a cabo una serie de test he podido comprobar que su grado de reconocimiento no es tan alto. Por tanto, le aconsejo que pruebe los degradados con Firefox, con y sin prefijo.

2. Los degradados de Mozilla Firefox

Se trata de la propiedad `-moz-linear-gradient()` la que permite realizar degradados lineales. Esta propiedad admite diversos argumentos.

- El primer argumento es la posición inicial, que puede tomar una palabra clave como valor (`top`, `right`, `bottom` o `left`), o bien un valor en píxeles. Si usted indica `top left`, el punto de partida será el borde superior izquierdo, equivalente a `0 0`. Dispone de un segundo valor facultativo, se trata del ángulo del degradado expresado en grados (`deg`), en gradiantes (`grads`) o en radianes (`rad`). Ejemplo: `23deg`.
- El segundo argumento es el color de partida, que puede ir seguido de la posición de partida expresada en %, en una escala de 0 a 100%.
- El tercer argumento es el color de llegada, que puede ir seguido de la posición de llegada expresada en %, en una escala de 0 a 100%.

Veamos un ejemplo (fíjese que en el ejemplo he insertado un salto de línea tras cada argumento para facilitar la lectura):

```
#degradado1 {
   position: absolute;
```

```
    top: 10px;
    left: 10px;
    width: 200px;
    height: 90px;
    background: -moz-linear-gradient(
       top left -23deg,
       rgb(30,87,153) 13%,
       rgb(237,230,40) 88%
    );
}
```

- el degradado comienza en la parte superior izquierda: `top left`.
- con un ángulo de -23 grados: `-23deg`.
- el primer color es un azul que ocupa el 13% de la totalidad del degradado:`rgb(30,87,153) 13%`.
- el color final es un amarillo que ocupa el 88 % de la totalidad del degradado:`rgb(237,230,40) 88%)`.

Así se visualiza en Firefox:

Veamos ahora un degradado con cuatro colores:

```
#degradado2 {
   position: absolute;
   top: 130px;
   left: 10px;
   width: 200px;
   height: 90px;
   background: -moz-linear-gradient(top,
      rgb(30,87,153) 0%,
      rgb(237,230,40) 30%,
      rgb(31,186,77) 55%,
      rgb(239,40,143) 100%);
}
```

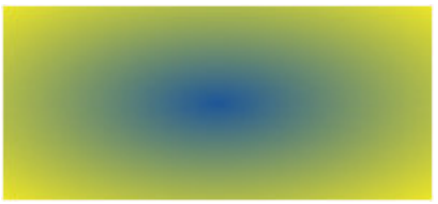
Este sería el resultado visual:

Los degradados radiales siguen el mismo principio, con la propiedad`-moz-linear-gradient()`. Dispone además de un nuevo argumento para indicar la forma del degradado radial: `circle` o`ellipse`.

Veamos un ejemplo:

```
background: -moz-radial-gradient(100px 45px, ellipse, rgb(30,87,153),
rgb(237,230,40));
```

Y la visualización en pantalla:



3. Les degradados para WebKit

Recordemos que los navegadores **Safari** y **Chrome** comparten el mismo motor de renderizado:**webkit**.

Esta sería la sintaxis para un degradado lineal:

```
background: -webkit-gradient(linear, left top, left bottom,
color-stop(0%,rgb(30,87,153)), color-stop(100%,rgb(237,230,40)));
```

- `-webkit-gradient()`: indica que se trata de un degradado.
- `linear`: un degradado lineal.
- `left top`: el punto de partida.
- `left bottom`: el punto de llegada.
- `color-stop(0%,rgb(30,87,153))`: primer color que comenzará a 0% del degradado y su descripción en rgb (azul).
- `color-stop(100%,rgb(237,230,40))`: primer color que comenzará a 0% del degradado y su descripción en rgb (amarillo).

Y el resultado visual:

4. Los degradados según el W3C

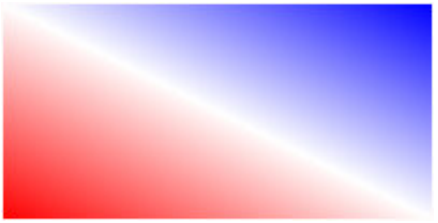
Los degradados del W3C (http://www.w3.org/TR/css3-images/#gradients) utilizan una sintaxis "oficial" que los navegadores aún no reconocen.

Ejemplo de un degradado lineal:

```
linear-gradient (to top right, red, white, blue)
```

Esto significa que queremos crear un degradado que vaya hacia arriba y hacia la derecha (`to top right`), con un primer color rojo (`red`), un segundo color blanco (`white`) y un último color azul (`blue`).

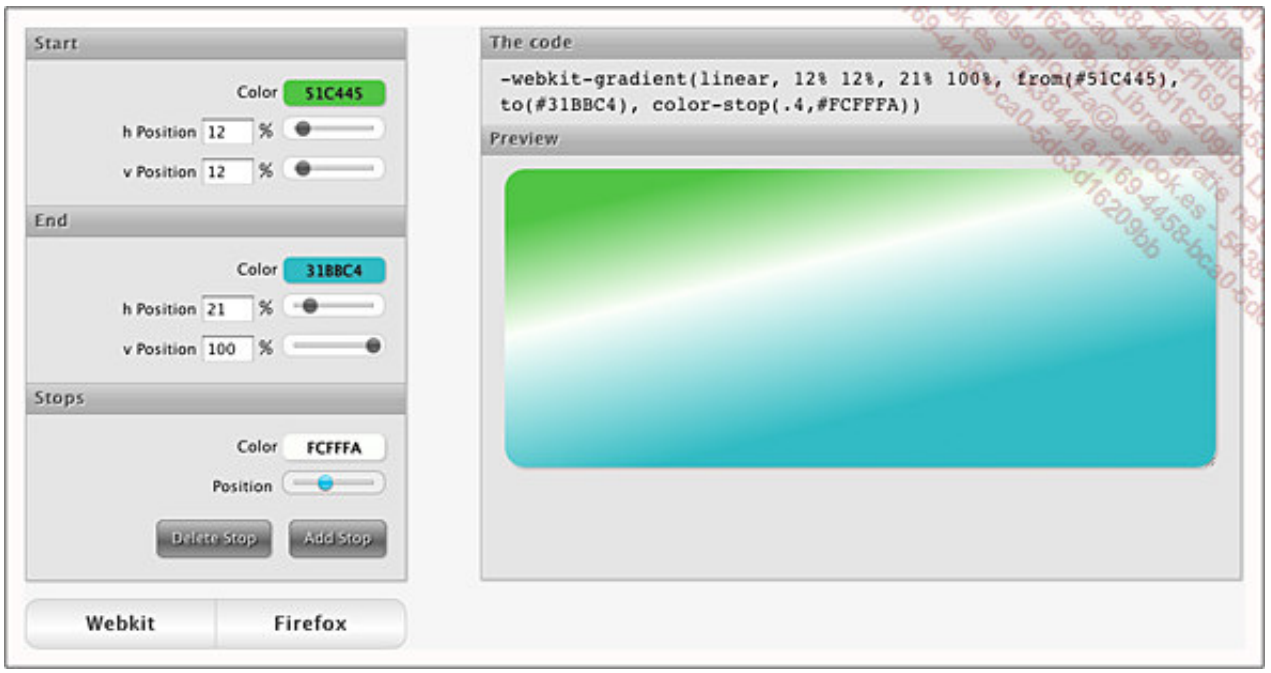
Este sería el resultado esperado:



5. Los generadores en línea

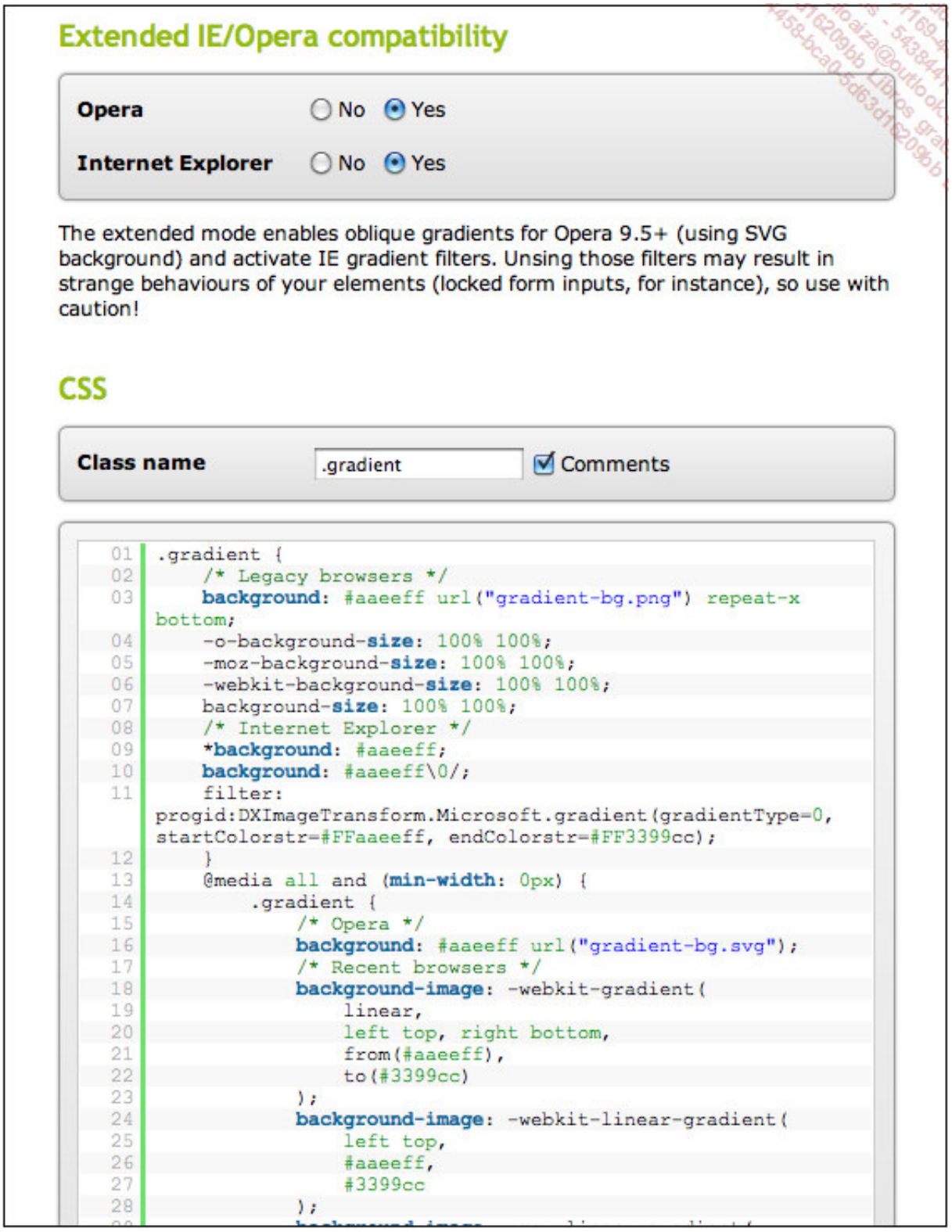
Como ha podido constatar, la situación actual es complicada. Existen multitud de sintaxis, una para cada navegador. Si desea crear degradados rápidamente, es preferible que use un generador en línea. Existen muchos. Veamos algunos de ellos.

WESTCIV (http://www.westciv.com/tools/gradients/) pone a su disposición una interfaz simple en la que podrá generar degradados optimizados para WebKit y Firefox.

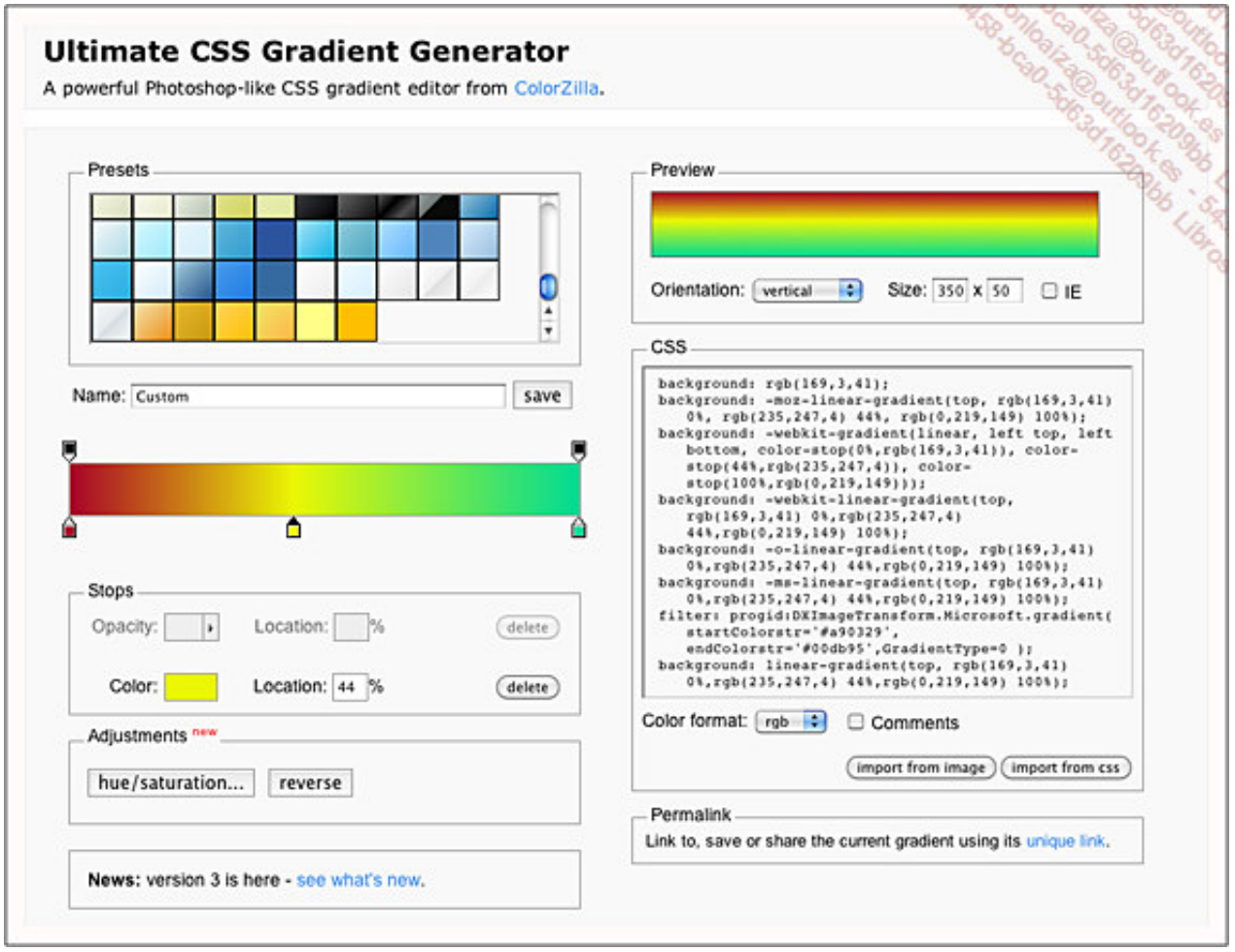


El sitio web **Display-inline** (http://www.display-inline.fr/projects/css-gradient/) presenta una interfaz muy completa.

A continuación podrá seleccionar la compatibilidad con los navegadores y se generará un código CSS bastante completo.



Y en último lugar, probablemente el generador preferido de los grafistas, **Ultimate CSS Gradient Generator** (http://www.colorzilla.com/gradient-editor/) que reutiliza la interfaz de Adobe Photoshop. ¡Una auténtica maravilla!


Ultimate CSS Gradient Generator
A powerful Photoshop-like CSS gradient editor from ColorZilla.
Presets
Name: Custom
save
Stops
Opacity:
Location:
%
delete
Color:
Location: 44 %
delete
Adjustments new
hue/saturation...
reverse
News: version 3 is here - see what's new.
Preview
Orientation: vertical
Size: 350 x 50
IE
CSS
background: rgb(169,3,41);
background: -moz-linear-gradient(top, rgb(169,3,41) 0%, rgb(235,247,4) 44%, rgb(0,219,149) 100%);
background: -webkit-gradient(linear, left top, left bottom, color-stop(0%,rgb(169,3,41)), color-stop(44%,rgb(235,247,4)), color-stop(100%,rgb(0,219,149)));
background: -webkit-linear-gradient(top, rgb(169,3,41) 0%,rgb(235,247,4) 44%,rgb(0,219,149) 100%);
background: -o-linear-gradient(top, rgb(169,3,41) 0%,rgb(235,247,4) 44%,rgb(0,219,149) 100%);
background: -ms-linear-gradient(top, rgb(169,3,41) 0%,rgb(235,247,4) 44%,rgb(0,219,149) 100%);
filter: progid:DXImageTransform.Microsoft.gradient(startColorstr='#a90329', endColorstr='#00db95',GradientType=0);
background: linear-gradient(top, rgb(169,3,41) 0%,rgb(235,247,4) 44%,rgb(0,219,149) 100%);
Color format: rgb
Comments
import from image
import from css
Permalink
Link to, save or share the current gradient using its unique link.

Los fondos múltiples

1. Funcionalidad

Con CSS2.1 solamente podemos aplicar una imagen como fondo de un elemento. Con CSS3 es posible aplicar varias imágenes, lo que nos ofrece posibilidades muy interesantes (http://www.w3.org/TR/css3-background/#layering).

2. Un ejemplo sencillo

Vamos a usar la propiedad `background`. La diferencia es simple: las distintas imágenes de fondo estarán separadas mediante una coma. En el ejemplo, he insertado un salto de línea tras cada imagen, para facilitar la lectura del código. Cada imagen corresponde a una cifra.

```
#imagenesMultiples {
   position: absolute;
   top:  10px;
   left: 10px;
   width: 150px;
   height: 80px;
   border: 1px solid #333;
   background: url("uno.png") top left no-repeat,
               url("dos.png") top right no-repeat,
               url("tres.png") bottom left no-repeat,
               url("cuatro.png") bottom right no-repeat;
}
```

Y el resultado visual:

Esta forma abreviada de la propiedad `background` es un "condensado" del uso de las propiedades:

- `background-image`: ruta de acceso, URL de la imagen.
- `background-position`: posición de la imagen indicada con palabras clave o con una unidad.
- `background-repeat`: repetición de la imagen.

Este sería el equivalente en sintaxis "larga":

```
#imagenesMultiples2 {
   position: absolute;
   top: 10px;
   left: 200px;
   width: 150px;
   height: 80px;
   border: 1px solid #333;
   background-image:url("uno.png"),
        url("dos.png"),
        url("tres.png"),
        url("cuatro.png");
```

```
    background-position: top left,
         top right,
         bottom left,
         bottom right;
    background-repeat: no-repeat;
}
```

Preste mucha atención, el orden de declaración de las imágenes indica su orden de aparición, independientemente de la posición. Analicemos la siguiente sintaxis para ilustrar ese orden de declaración y de visualización:

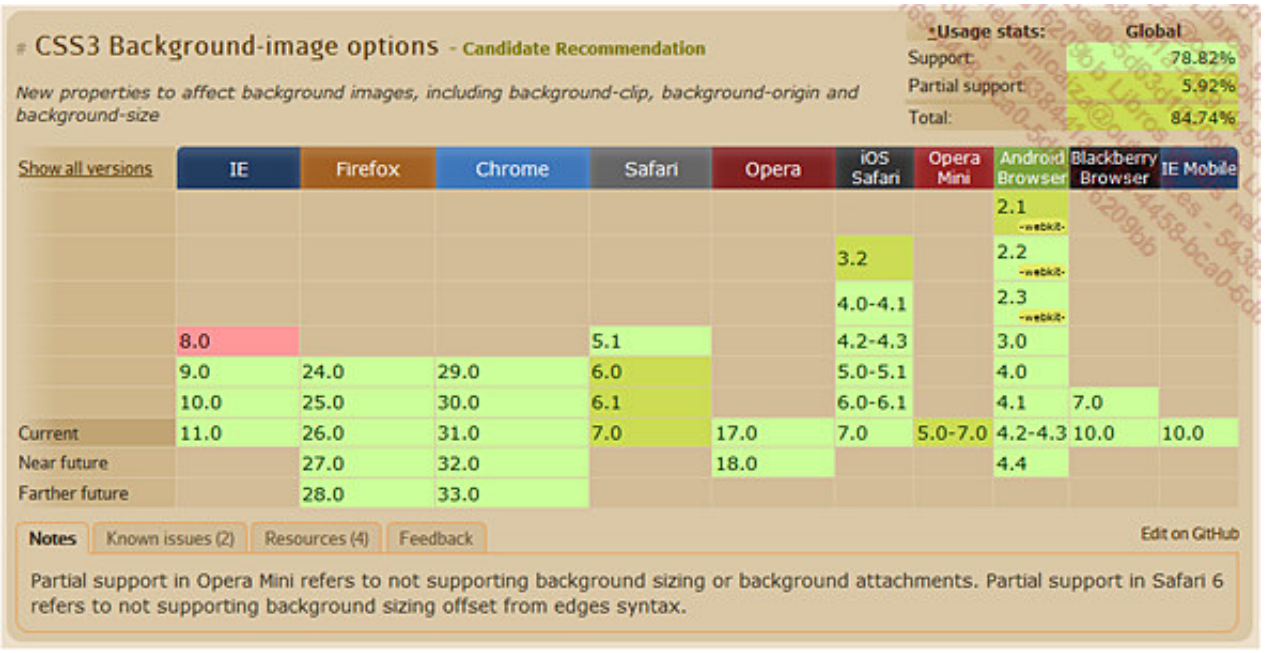
```
#imagenesMultiples3 {
    position: absolute;
    top: 10px;
    left: 380px;
    width: 150px;
    height: 80px;
    border: 1px solid #333;
    background: url("uno.png") top left no-repeat,
                url("tigre.jpg") top right no-repeat,
                url("tres.png") bottom left no-repeat,
                url("cuatro.png") bottom right no-repeat;
}
```

En el ejemplo, la fotografía del **tigre** se ha declarado en segundo lugar, tras la imagen del número **1**. Esa fotografía (que tiene las mismas dimensiones que la caja) va a ocultar las imágenes de las cifras**3** y **4**.

Otras propiedades del fondo

1. Los navegadores

Las nuevas propiedades (`background-origin`, `background-clip` y `background-size`) del fondo (http://www.w3.org/TR/css3-background/#background) son sólo parcialmente reconocidas por los navegadores. Por lo tanto, y como es habitual, resulta conveniente comprobarlas con y sin prefijo.



CSS3 Background-image options - Candidate Recommendation

New properties to affect background images, including background-clip, background-origin and background-size

*Usage stats: Global

Support: 78.82%

Partial support: 5.92%

Total: 84.74%

Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	IE Mobile
								2.1 -webkit-		
						3.2		2.2 -webkit-		
						4.0-4.1		2.3 -webkit-		
	8.0			5.1		4.2-4.3		3.0		
	9.0	24.0	29.0	6.0		5.0-5.1		4.0		
	10.0	25.0	30.0	6.1		6.0-6.1		4.1	7.0	
Current	11.0	26.0	31.0	7.0	17.0	7.0	5.0-7.0	4.2-4.3	10.0	10.0
Near future		27.0	32.0		18.0			4.4		
Farther future		28.0	33.0							

Notes | Known issues (2) | Resources (4) | Feedback | Edit on GitHub

Partial support in Opera Mini refers to not supporting background sizing or background attachments. Partial support in Safari 6 refers to not supporting background sizing offset from edges syntax.

2. La posición inicial de una imagen

La propiedad `background-origin` permite definir la posición inicial de una imagen de fondo de una caja. Esta propiedad admite tres valores:

- `padding-box`: la imagen se situará en el límite del margen de la caja. Se trata del valor por defecto.
- `border-box`: la imagen se situará en el límite del borde de la caja.
- `content-box`: la imagen se situará en el límite del contenido de la caja.

Por el momento deberá usar los prefijos propietarios y los valores propietarios (`border`,`padding` y `content`).

Veamos un ejemplo:

```
#posicion-inicial {
   width: 250px;
   height: 150px;
   border: 10px dotted; #333;
   padding: 10px;
   background: url("suricata.jpg")no-repeat;
   -webkit-background-origin: border;
   -moz-background-origin: border;
   background-clip: border-box;
}
```

Así se visualizará con el valor `border-box`:

Así se visualizará con el valor `padding-box`:

Así se visualizará con el valor `content-box`:

Fíjese que la imagen aparece recortada.

3. Recortar una imagen

La propiedad `background-clip` permite recortar una imagen de fondo. Esta propiedad utiliza los mismo valores de la propiedad `background-origin`, con las mismas especificaciones.

Veamos un ejemplo con un fondo de color:

```
#recortar {
   width: 250px;
   height: 150px;
   border: 10px dotted; #333;
   padding: 10px;
```

```
    background-color: lightblue;
    -webkit-background-clip: border;
    -moz-background-clip: border;
    background-clip: border-box;
}
```

Así se visualizará con el valor `border-box`. Se puede ver el fondo azul en el borde.

Así se visualizará con el valor `padding-box`. El fondo azul se aplica al contenido y al espaciado (`padding`), pero no al borde.

Así se visualizará con la valor `content-box`. El fondo azul se aplica solamente al contenido (`width`).

4. El tamaño del fondo

La propiedad `background-size` permite definir el tamaño de la imagen de fondo en función del elemento que la contenga. Los valores posibles son:

- Un porcentaje del tamaño del elemento "contenedor": un único valor indicará el mismo tamaño para el lado horizontal y el vertical. Dos valores indicarán el tamaño horizontal, con el primer valor, y el tamaño vertical, con el segundo.
- Un valor: un único valor indicará el tamaño horizontal, y el tamaño vertical se calculará proporcionalmente.
- Dos valores indicarán el tamaño horizontal, con el primer valor, y el tamaño vertical, con el segundo.
- La palabra clave `cover` permite redimensionar la imagen todo lo posible, `contain`redimensiona conservando las proporciones.

En el ejemplo se ha reducido la imagen al 50% del tamaño de la caja que la contiene:

```
#tamaño {
   width: 150px;
   height: 100px;
   border: 1px solid; #333;
   background: url("suricata.jpg") no-repeat;
   background-size: 50%;
}
```

Así se verá en la pantalla:

En este ejemplo, el tamaño es del 100% para las dos dimensiones, la imagen aparece deformada:

```
#tamaño{
   width: 150px;
   height: 50px;
   border: 1px solid; #333;
   background: url("suricata.jpg") no-repeat;
   background-size: 100% 100%;
}
```

Así se verá en la pantalla:

En este ejemplo, la imagen tiene un tamaño fijo, no proporcional:

```
#tamaño {
   width: 150px;
   height: 50px;
   border: 1px solid; #333;
   background: url("suricata.jpg") no-repeat;
   background-size: 100px 30px;
 }
```

Así se verá en la pantalla:

En este ejemplo, el valor `cover` permite redimensionar proporcionalmente la imagen en un sentido. En este caso se trata de una redimensión horizontal. La imagen aparece cortada verticalmente.

```
#tamaño{
   width: 150px;
   height: 50px;
   border: 1px solid; #333;
   background: url("suricata.jpg") no-repeat;
   background-size: cover;
}
```

Así se verá en la pantalla:

En este último ejemplo, el valor `contain` permite redimensionar proporcionalmente la imagen en un sentido que permita su publicación sin que aparezca cortada. En este caso se trata de una redimensión vertical. De este modo queda espacio libre horizontalmente.

```
#tamaño{
   width: 150px;
   height: 50px;
   border: 1px solid; #333;
   background: url("suricata.jpg") no-repeat;
   background-size: contain;
}
```

Así se verá en la pantalla:

Los generadores CSS3 en línea

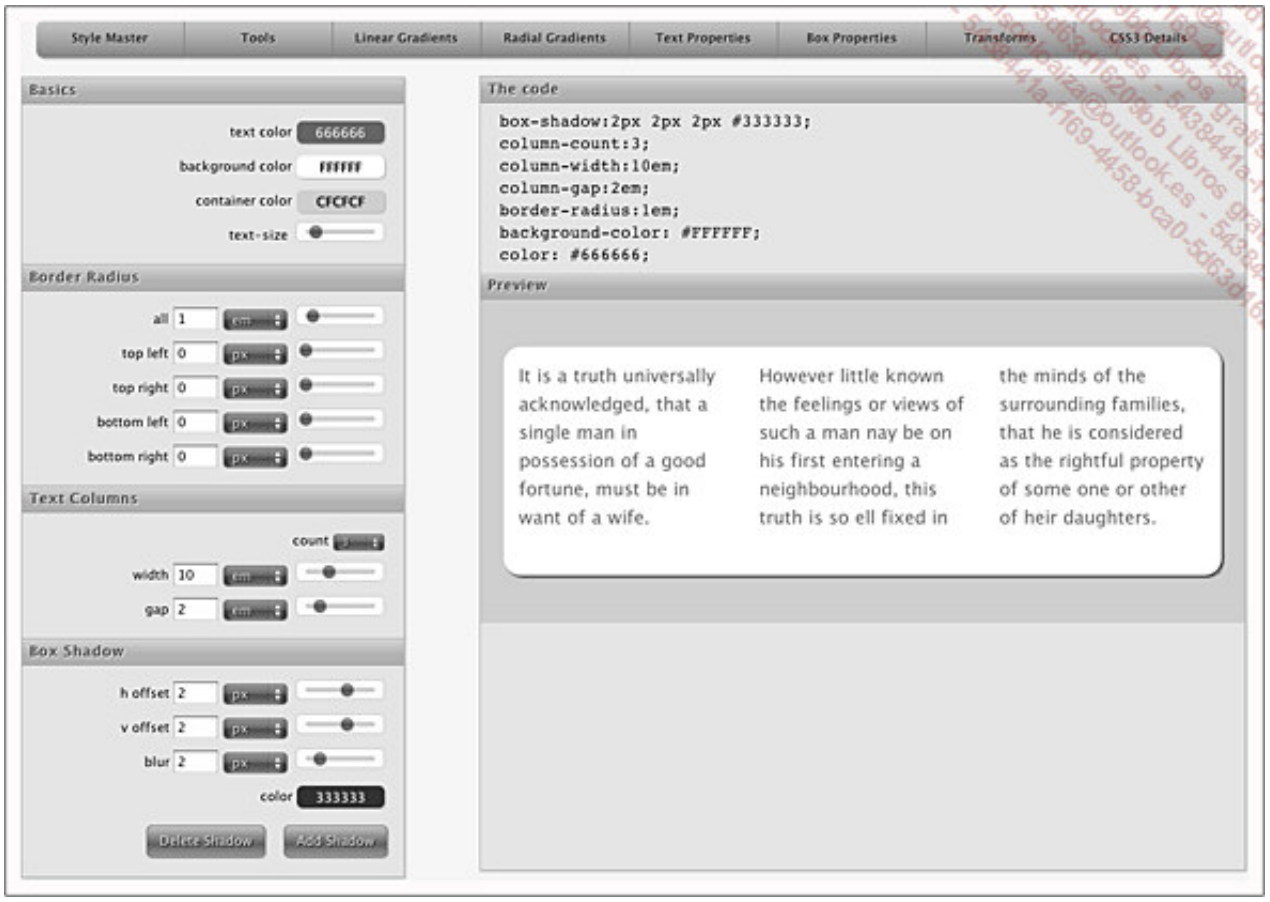
1. Funcionalidad

Ya hemos visto algunos generadores en línea para las nuevas propiedades de CSS3. Voy a presentarle ahora algunas herramientas "todo en uno", en las que encontrará prácticamente todas las propiedades que necesite.

2. WestCIV

Este sitio web (http://www.westciv.com/tools/box-properties/index.html) permite trabajar, en una única interfaz, con:

- degradados lineales y radiales,
- texto (sombra y columnas),
- cajas (color de fondo, bordes redondeados, columnas y sombras),
- transformaciones (que veremos en un próximo capítulo).



3. CSS 3.0 Maker

Este sitio web (http://www.css3maker.com/index.html), con una interfaz muy oscura, le permite trabajar con:

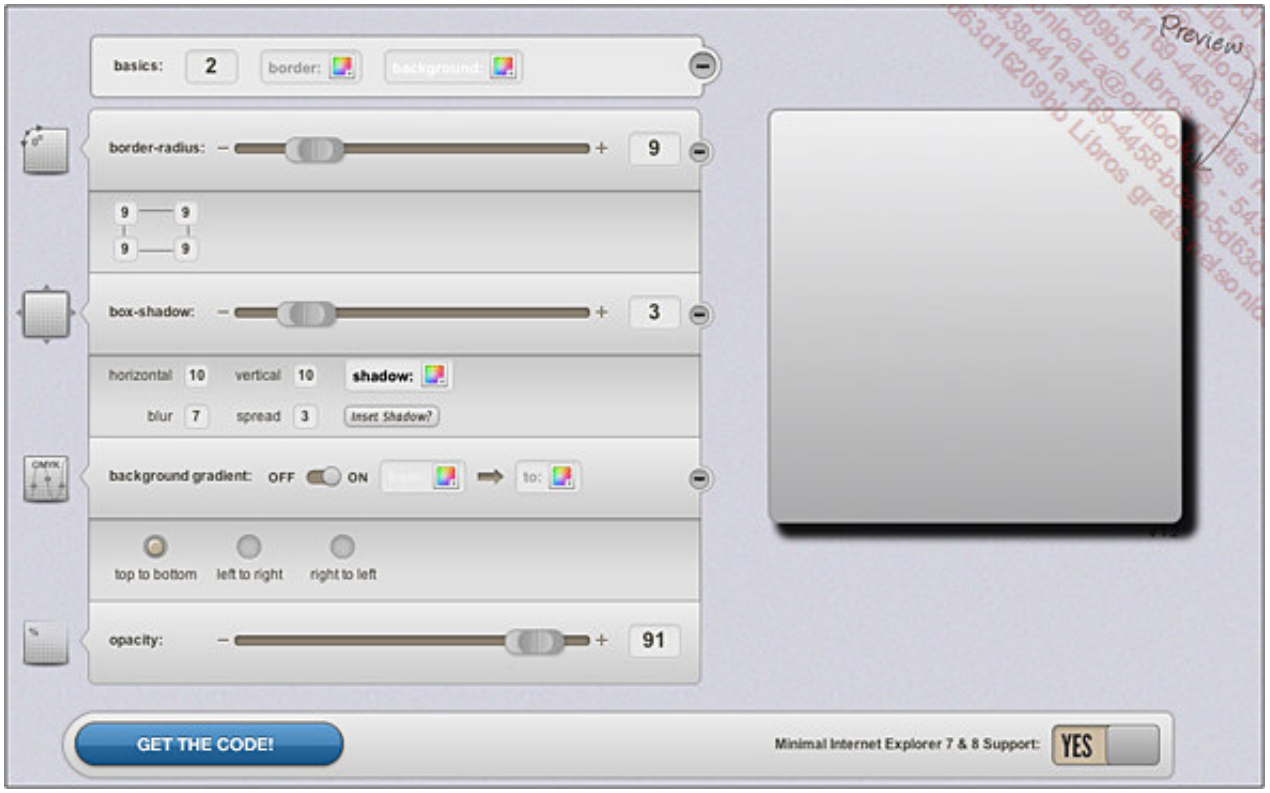
- bordes redondeados,
- degradados,

- transformaciones, animaciones y transiciones (que veremos en un próximo capítulo),
- el color del fondo en `rgba()`,
- la sombra del texto,
- la sombra de las cajas,
- la rotación del texto,
- la gestión de las fuentes incorporadas con `@font-face`.

4. CSS3 Generator

Este sitio web (http://www.css3.me/), con una interfaz bien diseñada, permite trabajar con:

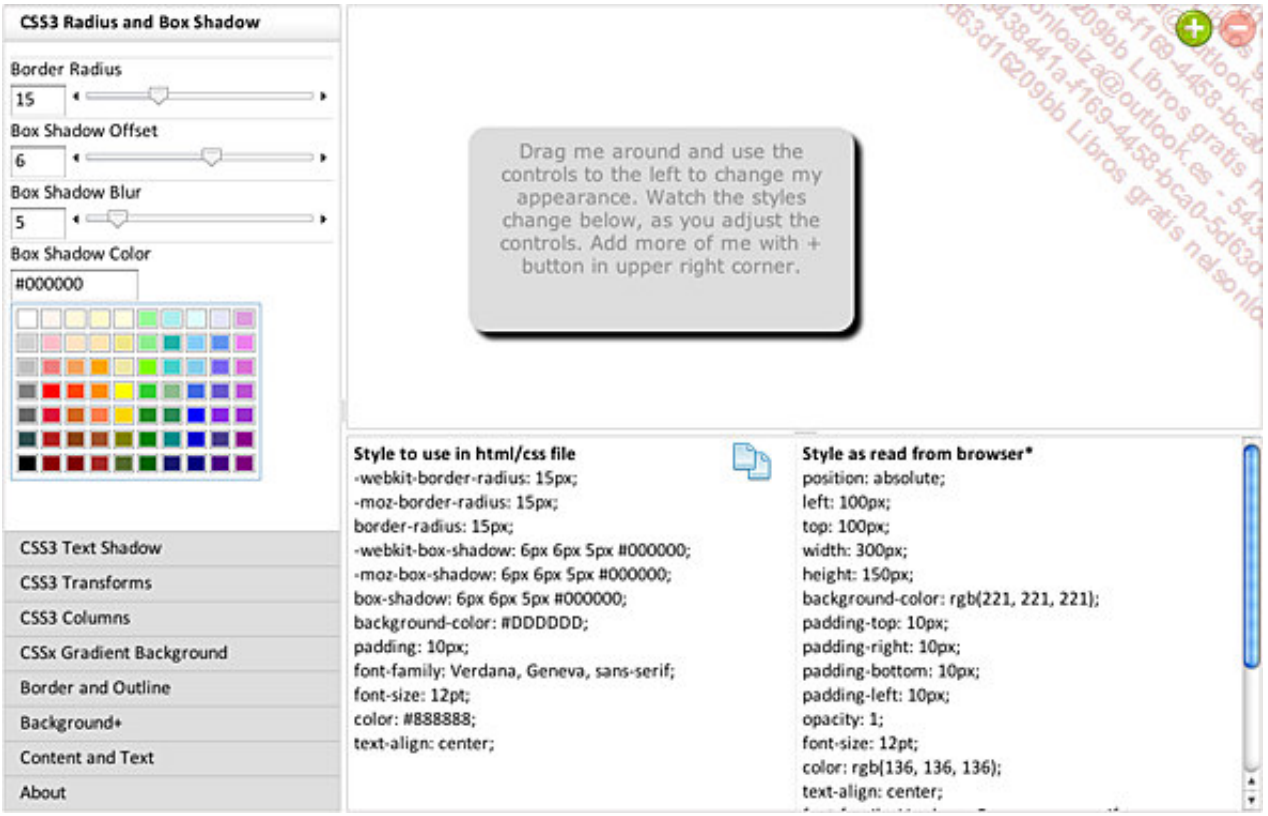
- bordes,
- sombras de cajas,
- degradados,
- transparencia.



5. CSS3 Playground

Este sitio web (http://css3.mikeplate.com/) con una interfaz minimalista permite trabajar con:

- bordes redondeados,
- sombras de cajas,
- sombras de texto,
- transformaciones,
- columnas de cajas,
- degradados,
- bordes y subrayado (`outline`),
- color y transparencia del fondo,
- texto (fuente, tamaño, alineación...).



6. CSS3 Generator

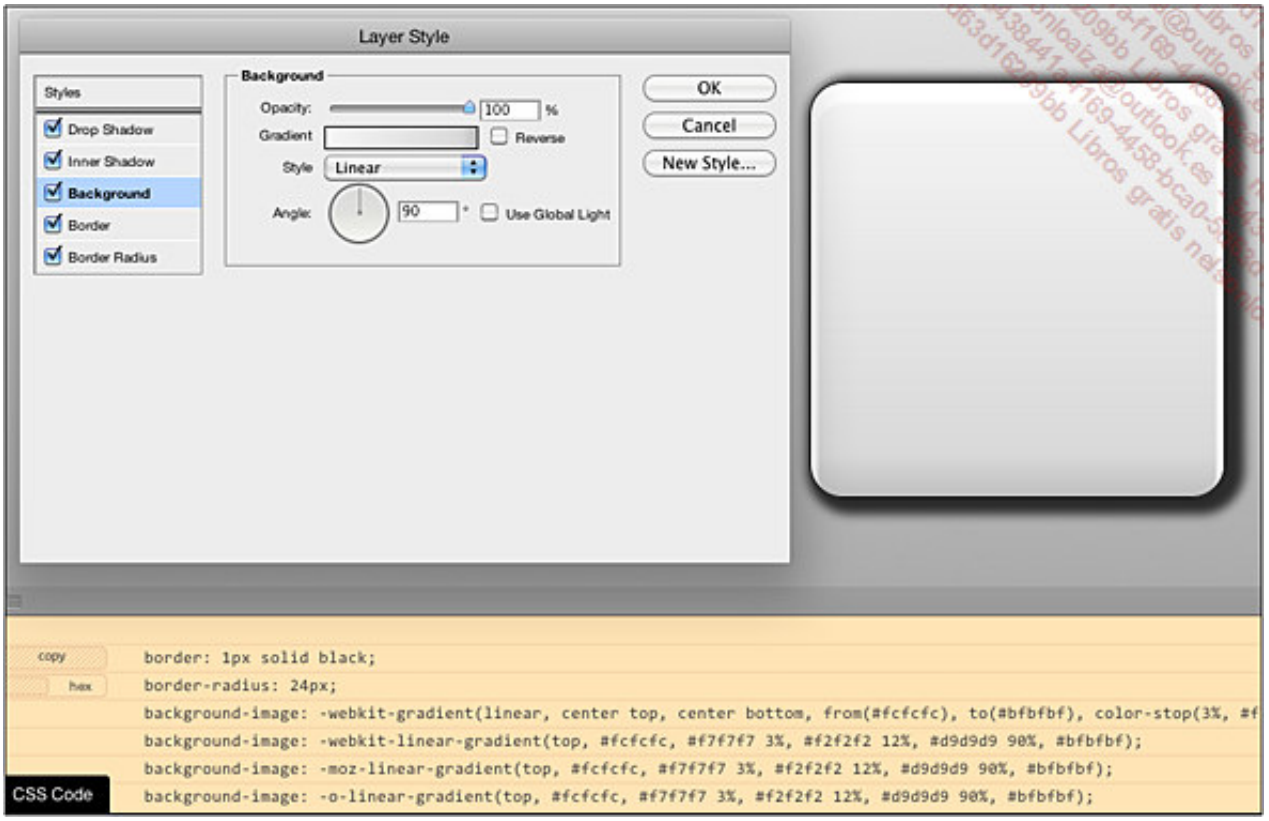
Este sitio web (http://css3generator.com/) permite trabajar con:

- bordes redondeados,
- sombras de cajas,
- sombras de texto,
- el color del fondo en `rgba()`,
- las fuentes con la regla `@font-face`,
- las columnas de las cajas,
- la redimensión de las cajas (`resize`),
- el modo de cálculo del tamaño de las cajas (`box-sizing`),
- el subrayado (`outline`),
- transiciones y transformaciones (que veremos en un próximo capítulo).

7. Layer Style

Este sitio web (http://layerstyles.org/builder.html), con una interfaz sobria pero desbordante de interactividad (podríamos creer que nos encontramos en Photoshop para Mac), nos permite trabajar con:

- sombras de cajas, externas e internas,
- degradados y transparencia,
- bordes,
- bordes redondeados.




```
copy
hex
border: 1px solid black;
border-radius: 24px;
background-image: -webkit-gradient(linear, center top, center bottom, from(#fcfcfc), to(#bfbfbf), color-stop(3%, #f
background-image: -webkit-linear-gradient(top, #fcfcfc, #f7f7f7 3%, #f2f2f2 12%, #d9d9d9 90%, #bfbfbf);
background-image: -moz-linear-gradient(top, #fcfcfc, #f7f7f7 3%, #f2f2f2 12%, #d9d9d9 90%, #bfbfbf);
background-image: -o-linear-gradient(top, #fcfcfc, #f7f7f7 3%, #f2f2f2 12%, #d9d9d9 90%, #bfbfbf);
```

CSS Code

Ejemplo de una tabla con bordes redondeados

1. Funcionalidad

Vamos a estudiar cómo se ha creado la tabla con bordes redondeados que propone **Red Team Design**: http://www.red-team-design.com/practical-css3-tables-with-rounded-corners

#	IMDB Top 10 Movies	Year
1	The Shawshank Redemption	1994
2	The Godfather	1972
3	The Godfather: Part II	1974
4	The Good, the Bad and the Ugly	1966
5	Pulp Fiction	1994
6	12 Angry Men	1957
7	Schindler's List	1993
8	One Flew Over the Cuckoo's Nest	1975
9	The Dark Knight	2008
10	The Lord of the Rings: The Return of the King	2003

2. La estructura de la tabla

La tabla (que utiliza una clase llamada `.bordered`) está estructurada de una manera muy clásica con`<table>`, `<thead>`, `<tr>`, `<th>` y `<td>`.

```
<table class="bordered">
    <thead>
    <tr>
        <th>#</th>
        <th>IMDB Top 10 Movies</th>
        <th>Year</th>
    </tr>
    </thead>
    <tr>
        <td>1</td>
        <td>The Shawshank Redemption</td>
        <td>1994</td>
    </tr>
    <tr>
        <td>2</td>
        <td>The Godfather</td>
        <td>1972</td>
    </tr>
    ...
```

```
</table>
```

Así se visualizará inicialmente:

#	IMDB Top 10 Movies	Year
1	The Shawshank Redemption	1994
2	The Godfather	1972
3	The Godfather: Part II	1974
4	The Good, the Bad and the Ugly	1966
5	Pulp Fiction	1994
6	12 Angry Men	1957
7	Schindler's List	1993
8	One Flew Over the Cuckoo's Nest	1975
9	The Dark Knight	2008
10	The Lord of the Rings: The Return of the King	2003

3. El diseño de la tabla

La tabla no deberá presentar espacios vacíos entre las celdas con borde:

```
table {
    border-spacing: 0;
    width: 500px;
}
```

La clase `.bordered` permite aplicar bordes redondeados y una ligera sombra:

```
.bordered {
    border: solid #ccc 1px;
    -moz-border-radius: 20px;
    -webkit-border-radius: 20px;
    border-radius: 20px;
    -webkit-box-shadow: 0 3px 3px #ccc;
    -moz-box-shadow: 0 3px 3px #ccc;
    box-shadow: 0 3px 3px #ccc;
}
```

El resultado visual:

#	IMDB Top 10 Movies	Year
1	The Shawshank Redemption	1994
2	The Godfather	1972
3	The Godfather: Part II	1974
4	The Good, the Bad and the Ugly	1966
5	Pulp Fiction	1994
6	12 Angry Men	1957
7	Schindler's List	1993
8	One Flew Over the Cuckoo's Nest	1975
9	The Dark Knight	2008
10	The Lord of the Rings: The Return of the King	2003

4. Las celdas de la tabla

Todas las celdas de la tabla (<th> y <td>) tienen un borde en los lados izquierdo y superior:

```
.bordered td, .bordered th {
    border-left: 1px solid #ccc;
    border-top: 1px solid #ccc;
    padding: 10px;
    text-align: left;
}
```

5. Las celdas del encabezado

Las celdas del encabezado <th> de la primera línea presentan un degradado de fondo y una sombra, al igual que el texto:

```
.bordered th {
    background-color: #dce9f9;
    background-image: -webkit-gradient(linear, left top, left
bottom, from(#ebf3fc), to(#dce9f9));
    background-image: -webkit-linear-gradient(top, #ebf3fc, #dce9f9);
    background-image: -moz-linear-gradient(top, #ebf3fc, #dce9f9);
    background-image: -ms-linear-gradient(top, #ebf3fc, #dce9f9);
    background-image: -o-linear-gradient(top, #ebf3fc, #dce9f9);
    background-image: linear-gradient(top, #ebf3fc, #dce9f9);
    -webkit-box-shadow: 0 1px 0 rgba(255,255,255,.8) inset;
    -moz-box-shadow:0 1px 0 rgba(255,255,255,.8) inset;
    box-shadow: 0 1px 0 rgba(255,255,255,.8) inset;
    border-top: none;
    text-shadow: 0 1px 0 rgba(255,255,255,.5);
}
```

El resultado visual:

#	IMDB Top 10 Movies	Year
1	The Shawshank Redemption	1994
2	The Godfather	1972
3	The Godfather: Part II	1974
4	The Good, the Bad and the Ugly	1966
5	Pulp Fiction	1994
6	12 Angry Men	1957
7	Schindler's List	1993
8	One Flew Over the Cuckoo's Nest	1975
9	The Dark Knight	2008
10	The Lord of the Rings: The Return of the King	2003

El degradado de la primera línea aún no aparece redondeado y quedan restos del borde en el lado izquierdo.

6. Celdas redondeadas

Para la primera línea de la tabla, no queremos que aparezca el borde de la primera celda `<td>` y la primera celda `<th>`.

```
.bordered td:first-child, .bordered th:first-child {
    border-left: none;
}
```

Ahora queremos que aparezcan redondeadas las primeras y las últimas celdas de la primera y la última línea. Utilizaremos las pseudo-clases `:first-child` y `:last-child` para las celdas del encabezado `<th>`, las líneas `<tr>` y las celdas `<td>`.

```
.bordered th:first-child {
    -moz-border-radius: 20px 0 0 0;
    -webkit-border-radius: 20px 0 0 0;
    border-radius: 20px 0 0 0;
}

.bordered th:last-child {
    -moz-border-radius: 0 20px 0 0;
    -webkit-border-radius: 0 20px 0 0;
    border-radius: 0 20px 0 0;
}

.bordered tr:last-child td:first-child {
    -moz-border-radius: 0 0 0 20px;
    -webkit-border-radius: 0 0 0 20px;
```

```
    border-radius: 0 0 0 20px;
}

.bordered tr:last-child td:last-child {
    -moz-border-radius: 0 0 20px 0;
    -webkit-border-radius: 0 0 20px 0;
    border-radius: 0 0 20px 0;
}
```

#	IMDB Top 10 Movies	Year
1	The Shawshank Redemption	1994
2	The Godfather	1972
3	The Godfather: Part II	1974
4	The Good, the Bad and the Ugly	1966
5	Pulp Fiction	1994
6	12 Angry Men	1957
7	Schindler's List	1993
8	One Flew Over the Cuckoo's Nest	1975
9	The Dark Knight	2008
10	The Lord of the Rings: The Return of the King	2003

7. La activación de las líneas

Al pasar el ratón sobre las líneas, cambiará el color del fondo. Esta propiedad se consigue con la pseudo-clase `:hover`.

```
.bordered tr:hover {
    background: #fbf8e9;
}
```

Este sería el resultado:

#	IMDB Top 10 Movies	Year
1	The Shawshank Redemption	1994
2	The Godfather	1972
3	The Godfather: Part II	1974
4	The Good, the Bad and the Ugly	1966
5	Pulp Fiction	1994
6	12 Angry Men	1957
7	Schindler's List	1993
8	One Flew Over the Cuckoo's Nest	1975
9	The Dark Knight	2008
10	The Lord of the Rings: The Return of the King	2003

Ejemplo de sombra para una página

Este es un truco muy sencillo que le permitirá aportar profundidad a sus páginas web:http://playground.genelocklin.com/depth/

La técnica consiste en aplicar la pseudo-clase `:before` a `body`.

El estilo CSS:

```
body:before {
   /* Contenido vacío*/
   content: "";
   /* Posición*/
   position: fixed;
   z-index: 100;
   top: -10px;

   left: -10px;
   /* Ancho y alto */
   width: 110%;
   height: 10px;
   /* Sombra */
   -webkit-box-shadow: 0px 0px 10px rgba(0,0,0,.8);
   -moz-box-shadow: 0px 0px 10px rgba(0,0,0,.8);
   -ms-box-shadow: 0px 0px 10px rgba(0,0,0,.8);
   -o-box-shadow: 0px 0px 10px rgba(0,0,0,.8);
   box-shadow: 0px 0px 10px rgba(0,0,0,.8);
}
```

No hay ningún contenido: `content: ""`, pero para el navegador, sí que hay un elemento, aunque esté vacío. Por lo tanto, no hay nada que mostrar.

Ese contenido vacío está colado en una posición fija, para que no sea desplazado cuando se use la barra de desplazamiento. Se ha colocado a -10 píxeles, que será el tamaño del difuminado.

```
position: fixed;
z-index: 100;
top: -10px;
left: -10px;
```

El ancho es mayor que el de la página y la altura es de 10 píxeles.

```
width: 110%;
height: 10px;
```

La sombra se ha colocado a 0 píxeles horizontal y verticalmente, el tamaño del difuminado es de 10 píxeles y se ha usado el color negro, con una ligera transparencia:

```
box-shadow: 0px 10px rgba(0,0,0,.8);
```

Así se visualizará sobre el fondo blanco de la página:

Los elementos de texto obsoletos

1. La especificación del HTML5

En la especificación de HTML5, determinados elementos se consideran "obsoletos". Un elemento obsoleto es un elemento que no ha sido declarado en la especificación de HTML5. Sin embargo, como el HTML5 debe ser compatible con las versiones anteriores, los elementos HTML obsoletos seguirán siendo interpretados correctamente por los navegadores.

Una página web que use elementos HTML obsoletos se visualizará correctamente, aunque los motores de validación la considerarán como "no conforme".

Esta es la URL de los elementos obsoletos del sitio web del W3C:http://www.w3.org/TR/html5/obsolete.html

2. Los elementos en desuso o mal utilizados

El elemento `acronym` para los acrónimos se ha declarado obsoleto, y se prefiere en su lugar la abreviatura `abbr`. La diferencia práctica entre ambos es que un acrónimo se pronuncia como una palabra (ejemplos: LASER, OTAN...).

3. El formato del texto

Los elementos `basefont`, `font`, `big`, `center`, `strike`, `tt` y `u` han quedado obsoletos... desde hace tiempo, y han sido remplazados por los estilos CSS, que son mucho más eficaces.

Los elementos de texto redefinidos

1. La especificación del HTML5

El HTML5 ha redefinido la función de determinados elementos HTML, en lugar de declararlos obsoletos.

2. El elemento <b>

El elemento `b` permitía mostrar en la pantalla una palabra en negrita, sin que hubiese una connotación semántica. El elemento `strong` lo remplazó rápidamente, aportando una noción semántica y una utilidad para los sintetizadores vocales, en las páginas conformes a las normas de accesibilidad web.

En HTML5, el elemento `b` indica que una parte del texto se diferencia visualmente del resto, pero sin que esto tenga mayor importancia. Si no es ese el caso, debemos utilizar `strong`.

Así es como se define el elemento `b` en la especificación del HTML5: "*The b element represents a span of text to which attention is being drawn for utilitarian purposes without conveying any extra importance and with no implication of an alternate voice or mood, such as key words in a document abstract, product names in a review, actionable words in interactive text-driven software, or an article lede.*"

En definitiva, si lo que desea hacer es, simplemente, poner de relieve visualmente una palabra usando la negrita, y nada más, puede usar el elemento `b`.

3. El elemento

El elemento `strong` ha conocido una ligera evolución. Hasta ahora se había usado para indicar un**mayor énfasis** y para la síntesis vocal. Ahora sirve para indicar una **mayor importancia**. Debemos reconocer que la diferencia es bastante sutil.

4. El elemento <i>

El elemento `i` permitía presentar en la pantalla una palabra en cursiva, sin que hubiese una connotación semántica. Para mejorar la semántica y la accesibilidad, utilice el elemento `em`.

En HTML5, utilice el elemento `i` para señalar las palabras que no suele usar habitualmente (palabras técnicas, palabras en un idioma extranjero...) y para indicar que está usando un "tono diferente" al del resto del texto.

Así es como se define el elemento `i` en la especificación del HTML5: "*The i element represents a span of text in an alternate voice or mood, or otherwise offset from the normal prose in a manner indicating a different quality of text, such as a taxonomic designation, a technical term, an idiomatic phrase from another language, a thought, or a ship name in Western texts.*"

5. El elemento

El elemento `em` sirve para indicar que un texto debe leerse con énfasis.

6. El elemento <small>

Este elemento se usaba para escribir un texto con un tamaño de caracteres menor al del resto del texto. En HTML5, el elemento `small` se usa para presentar un texto "legal" con letra pequeña (restricciones, advertencias, copyright...).

7. El elemento <cite>

El elemento `cite` servía para señalar una citación o una referencia a una fuente externa. Ahora podrá usarlo para indicar el título de cualquier "obra creativa" (libro, artículo, ensayo, canción, película...), pero no, al menos en principio, para indicar el nombre de una persona, ya que una persona no es una obra. Este es un extracto de su definición: "*The cite element represents the title of a work (e.g. a book, a paper, an essay, a poem, a score, a song, a script, a film, a TV show, a game, a sculpture, a painting, a theatre production, a play, an opera, a musical, an exhibition, a legal case report, etc). A person's name is not the title of a work*."

8. El elemento <dl>

En HTML 4, el elemento `dl` permitía introducir listas de definiciones compuestas por los pares "término - definición del término". En HTML5 su uso se amplía a cualquier lista de asociaciones formada por cero a varios grupos "nombre - valor".

Así es como se define el elemento `dl` en la especificación del HTML5: "*The dl element represents an association list consisting of zero or more name-value groups (a description list).*"

De este modo se consigue ampliar el uso de `dl` a cualquier tipo de lista.

9. El elemento

En HTML 4.01, el atributo `start` del elemento `ol` había quedado obsoleto. En HTML5, `start`vuelve a ser un atributo estándar.

Este presenta además un nuevo atributo boleano, `reversed`, que permite invertir el sentido de visualización de una lista.

10. El elemento <hr>

En HTML 4, el elemento `hr` había visto reducida su utilidad y se usaba para representar una línea horizontal. En HTML5, `hr` se usa para señalar un cambio de tema en un párrafo, un corte en la escena de una historia... Así es como se define el elemento `hr` en la especificación del HTML5: "*The hr element represents a paragraph-level thematic break, e.g. a scene change in a story, or a transition to another topic within a section of a reference book*."

11. El elemento <a>

El elemento `a` constituye la esencia misma de Internet. Es gracias a este elemento que podemos enlazar nuestras páginas web mediante vínculos hipertextuales. En HTML5, este elemento nos permite crear "grupos" de vínculos con varios elementos de tipo bloque.

Veamos un ejemplo válido en HTML5:

```
<a href="mipagina.html">
   <h3>Mis ofertas</h3>
   <p>enero</p>
   <p>febrero</p>
</a>
```

Recuerde que no es posible insertar un elemento `a` dentro de otro elemento `a`.

Nuevos formatos para el texto

1. Las columnas

Esta novedad bastante útil del CSS3 nos ofrece la posibilidad de maquetar un texto en columnas. Existe un módulo específico del CSS3 para esta funcionalidad: **CSS Multi-column Layout Module** (http://www.w3.org/TR/css3-multicol/) que se encuentra en la fase de **Candidate Recommendation** del 12 de abril de 2011. Por el momento, todavía estamos lejos de alcanzar la compatibilidad con los navegadores, solamente Firefox, Safari y Chrome reconocen parcialmente estas propiedades.

La primera propiedad que podemos aplicar es, simplemente, el ancho de las columnas y su nombre.

- La propiedad `column-count` permite indicar cuántas columnas queremos crear.
- La propiedad `column-width` permite definir el ancho de cada columna.
- La propiedad `column-gap` permite indicar el espacio entre cada columna (la maqueta, en publicación impresa).

Tendremos que usar los prefijos propietarios de los navegadores para usar estas propiedades.

En el siguiente ejemplo se ha dividido en columnas una caja `<div>`.

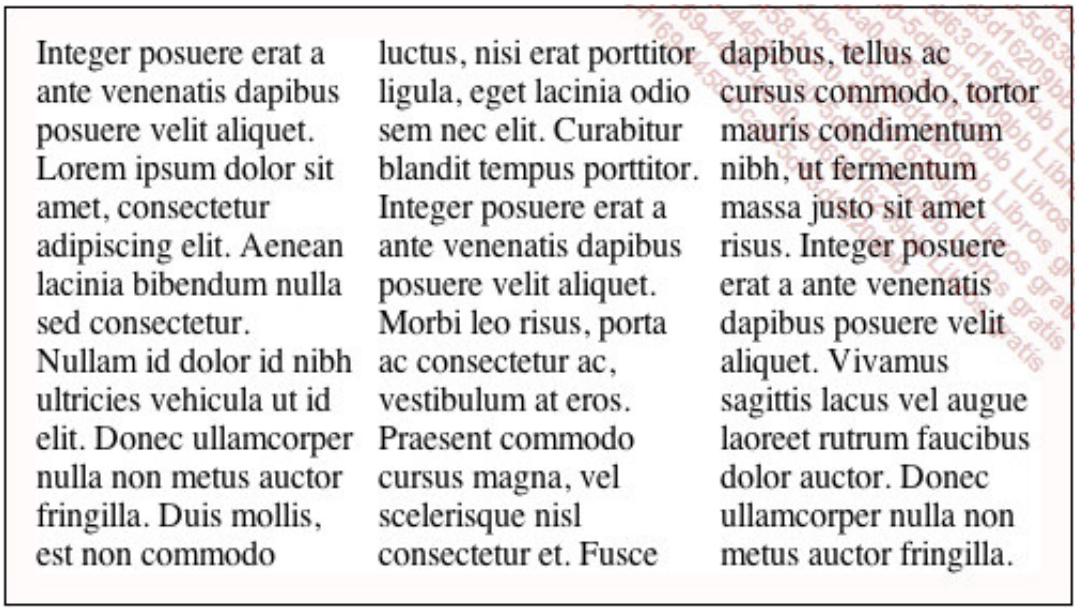
El estilo CSS:

```
#columnas3 {
   width: 470px;
   -moz-column-count: 3;
   -moz-column-width: 150px;
   -moz-column-gap: 10px;
   -webkit-column-count: 3;
   -webkit-column-width: 150px;
   -webkit-column-gap: 10px;
   column-count: 3;
   column-width: 150px;
   column-gap: 10px;
}
```

Este es el código HTML de la caja `<div>`:

```
<div id="columnas3">
   <p>Integer posuere erat...</p>
</div>
```

El resultado visual:



Si lo prefiere, puede usar la propiedad `columns`, que permite tener una sintaxis más corta, ya que reúne, en este orden, las propiedades `column-width` y `column-count`.

Ejemplo: `columns : 150px 3;`

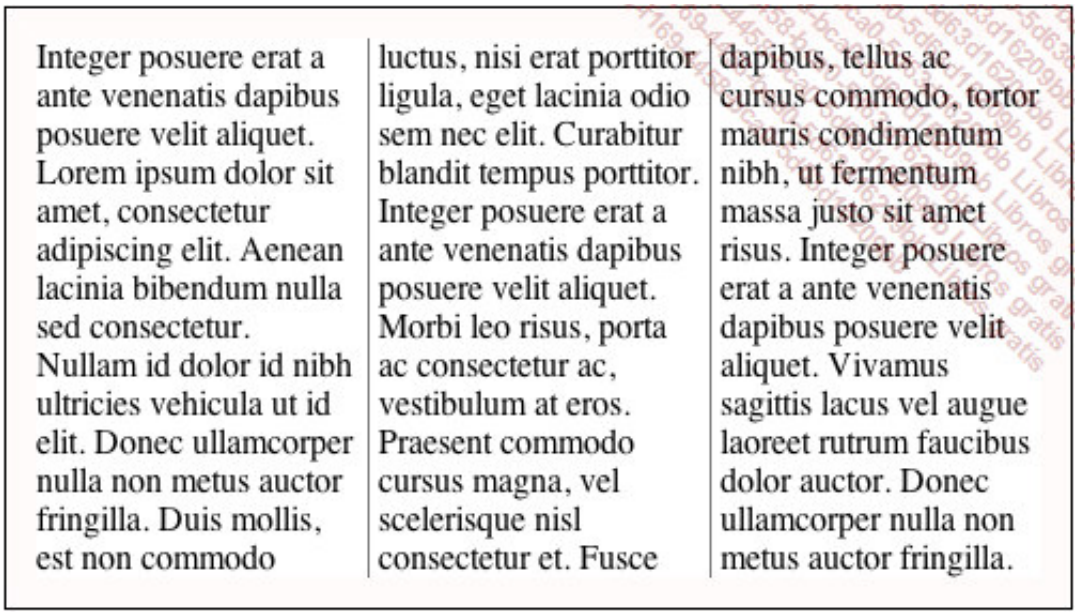
Podemos definir una línea vertical (de separación) entre las columnas, que se insertará en medio del espacio entre dichas columnas. Disponemos de las propiedades:

- `column-rule-width`: define el grosor de la línea de separación.
- `column-rule-style`: define el tipo o el estilo de la línea de separación.
- `column-rule-color`: define el color de la línea de separación.
- `column-rule`: es la sintaxis abreviada para definir, en este orden, el grosor, el estilo y el color.

Retomemos el ejemplo anterior, con la sintaxis abreviada:

```
#columnas3 {
   width: 470px;
   -moz-column-count: 3;
   -moz-column-width: 150px;
   -moz-column-gap: 10px;
   -moz-column-rule: 1px solid grey;
   -webkit-column-count: 3;
   -webkit-column-width: 150px;
   -webkit-column-gap: 10px;
   -webkit-column-rule: 1px solid grey;
   column-count: 3;
   column-width: 150px;
   column-gap: 10px;
   column-rule: 1px solid grey;
}
```

Se visualizará así:



Otras propiedades muy interesantes, como la gestión del salto de columna (`break-before`,`break-after` y `break-inside`) aún no son gestionadas por los navegadores, pero ofrecen interesantes perspectivas para la maquetación de tipo "revista".

En Internet encontrará multitud de herramientas en línea con las que podrá generar todo el código necesario para crear los estilos CSS3, que le permitirán presentar el texto en columnas.

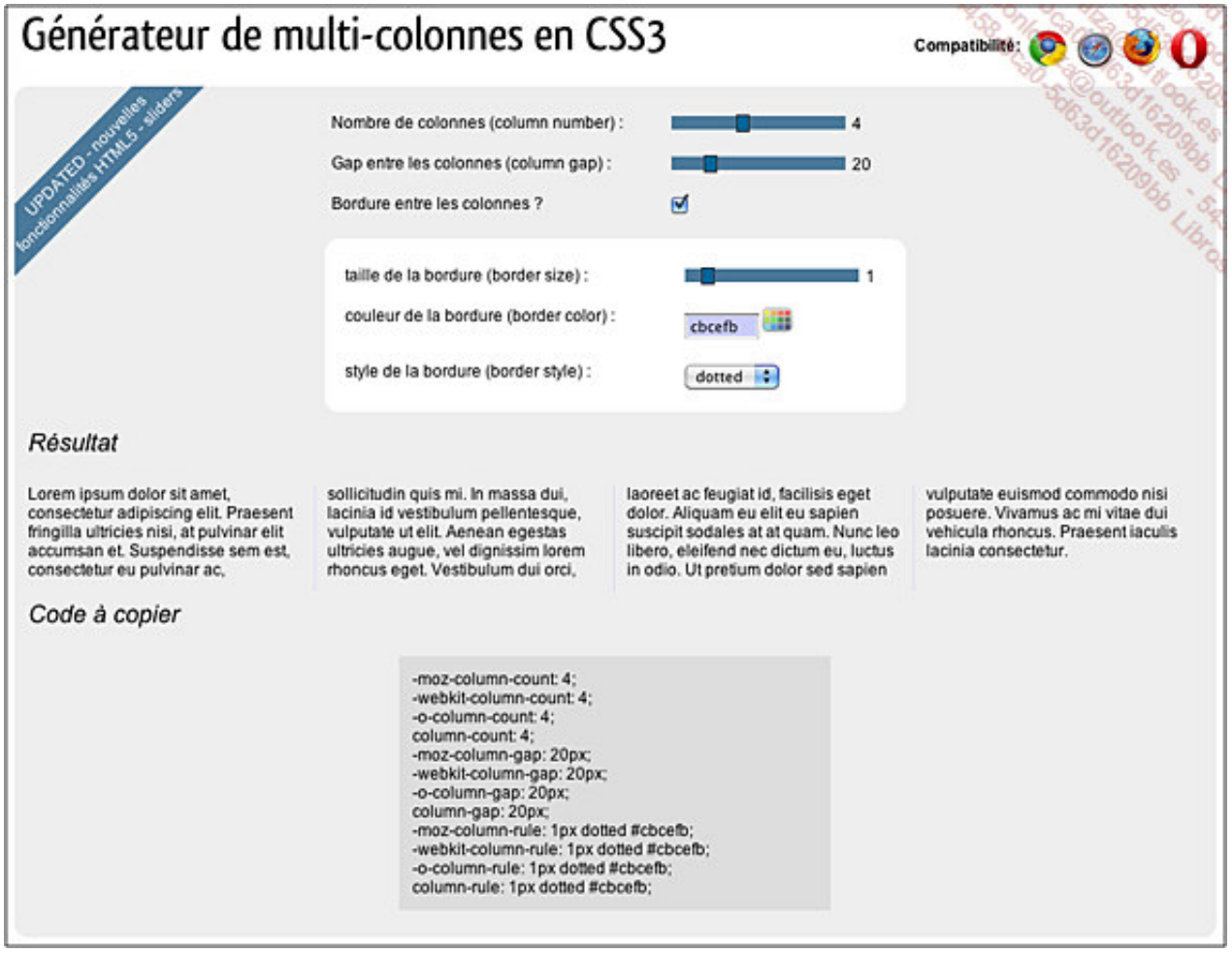
Yo le recomiendo **CSS3 Generator** (http://css3generator.com/). En el menú desplegable **Choose Something**, seleccione **Multiple Columns**.

En el campo **# of Columns**, indique el número de columnas que desee.

En el campo **Column Gap**, indique el espacio entre las columnas que desee aplicar.

En la zona **Your Code**, aparecerá el código compatible con **Firefox**, **Chrome** y **Safari**.

El sitio web **Dji** (del programador francés Jérôme Debray) nos propone multitud de herramientas en línea, incluido un generador de columnas (http://www.debray-jerome.fr/outils/Generateur-de-multi-colonnes-en-css3.html). La interfaz está en francés, pero es muy eficaz y fácil de usar. El código obtenido es compatible con **Chrome**, **Safari**, **Firefox** y **Opera**.



2. Las palabras cortadas

Los estilos CSS3 relacionados con el texto (http://www.w3.org/TR/css3-text/) nos proponen algunas novedades interesantes a la hora de dividir las palabras.

La propiedad `overflow-wrap` (antigua `word-wrap`) permite forzar que se seccione una palabra muy larga, cuando su longitud sea superior al ancho del elemento que la contenga. Si no se usa esta propiedad, la palabra demasiado larga "se saldrá" de su contenedor.

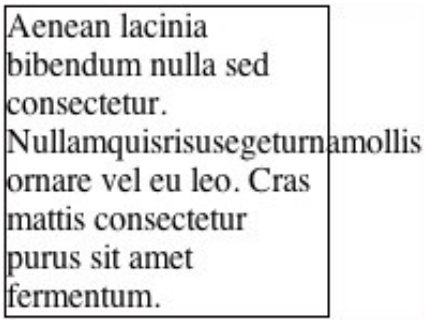
El estilo CSS:

```
.peque {
   width: 150px;
   border: solid 1px black;
   padding: 0;
 }
```

Este sería el código HTML:

```
<div>
   <p class="peque">Aenean lacinia bibendum nulla sed
consectetur. Nullamquisrisusegeturnamollis ornare vel eu leo.
Cras mattis consectetur purus sit amet fermentum.</p>
</div>
```

El resultado visual:

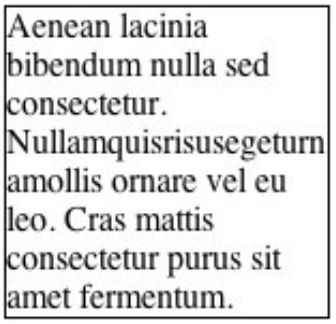


Apliquemos ahora la propiedad `overflow-wrap`, con su antigua sintaxis `word-wrap`, ya que los navegadores aún no reconoce la nueva. Los valores posibles son:

- `normal`: el corte se realiza en aquellos lugares en los que se haya autorizado la división.
- `hyphenate`: el corte se realiza en aquellos lugares en los que sea posible aplicar la división con guión.
- `break-word`: el corte se realiza de manera arbitraria, en función del ancho del contenido. Este es por el momento el único valor que reconocen los navegadores.

```
.peque {
   width: 150px;
   border: solid 1px black;
   padding: 0;
   word-wrap: break-word;
}
```

Este es el resultado obtenido:



Puede resultar interesante aplicarlo a las columnas.

Las fuentes tipográficas

1. El uso de las tipografías

La tipografía ha sido desde siempre el "pariente pobre" del diseño web. Siempre hemos tenido que limitarnos a las "familias genéricas", que eran las únicas que nos permitían obtener una visualización casi idéntica en los diferentes navegadores.

Teníamos:

- la familia de las tipografías sin serifa: "`arial`, `helvetica`, `sans-serif`".
- la familia de las tipografías con serifa: "`timesnew roman`, `times`, `serif`".
- la familia de las tipografías de ancho fijo: "`courier new`, `courier`, `mono`".

De esas tres listas, el navegador utiliza la primera tipografía que encuentre instalada en el ordenador del usuario.

Más tarde, Matthew Carter, a petición de Microsoft, diseñó dos tipografías creadas específicamente para facilitar la lectura en la pantalla: la tipografía sin serifa **Verdana** y la tipografía con serifa**Georgia**. Esas dos tipografías libres de derechos se encuentran disponibles en la gran mayoría de las plataformas informáticas actuales.

Debemos decir, sin embargo, que las posibilidades de elección son bastante limitadas.

2. Importar las fuentes tipográficas

Con los estilos CSS3 y la regla `@font-face` nosotros podemos "importar" las fuentes tipográficas en nuestra página web. De este modo, ya no estaremos limitados al uso de las fuentes genéricas.

¡Estupendo! Pero no se olvide de estos tres elementos fundamentales:

- La mayoría de las fuentes "profesionales" están sometidas a derechos de uso y de difusión.
- Cuando usted "importe" una fuente tipográfica, ¡incorporará a sus archivos la totalidad de los caracteres de dicha fuente! Esto podría sobrecargar en exceso sus páginas web.
- Por lo general, el *antialiasing* (que evita que los caracteres tengan un aspecto dentado) no se aplica a las páginas web.

3. La compatibilidad con los navegadores

Ahora tendremos que ocuparnos del problema de la compatibilidad (¡para no variar!) con las diferentes versiones de los navegadores.

Estos son los formatos de las fuentes tipográficas reconocidas por los navegadores:

- **TrueType**: extensión **.ttf**.
- **OpenType**: extensión **.otf**.
- **Web Open Font**: extensión **.woff**.
- **SVG Font**: extensión **.svg** y **.svgz**.
- **Embed OpenType**: extensión **.eot**. Atención, se trata de un formato propietario de Microsoft.

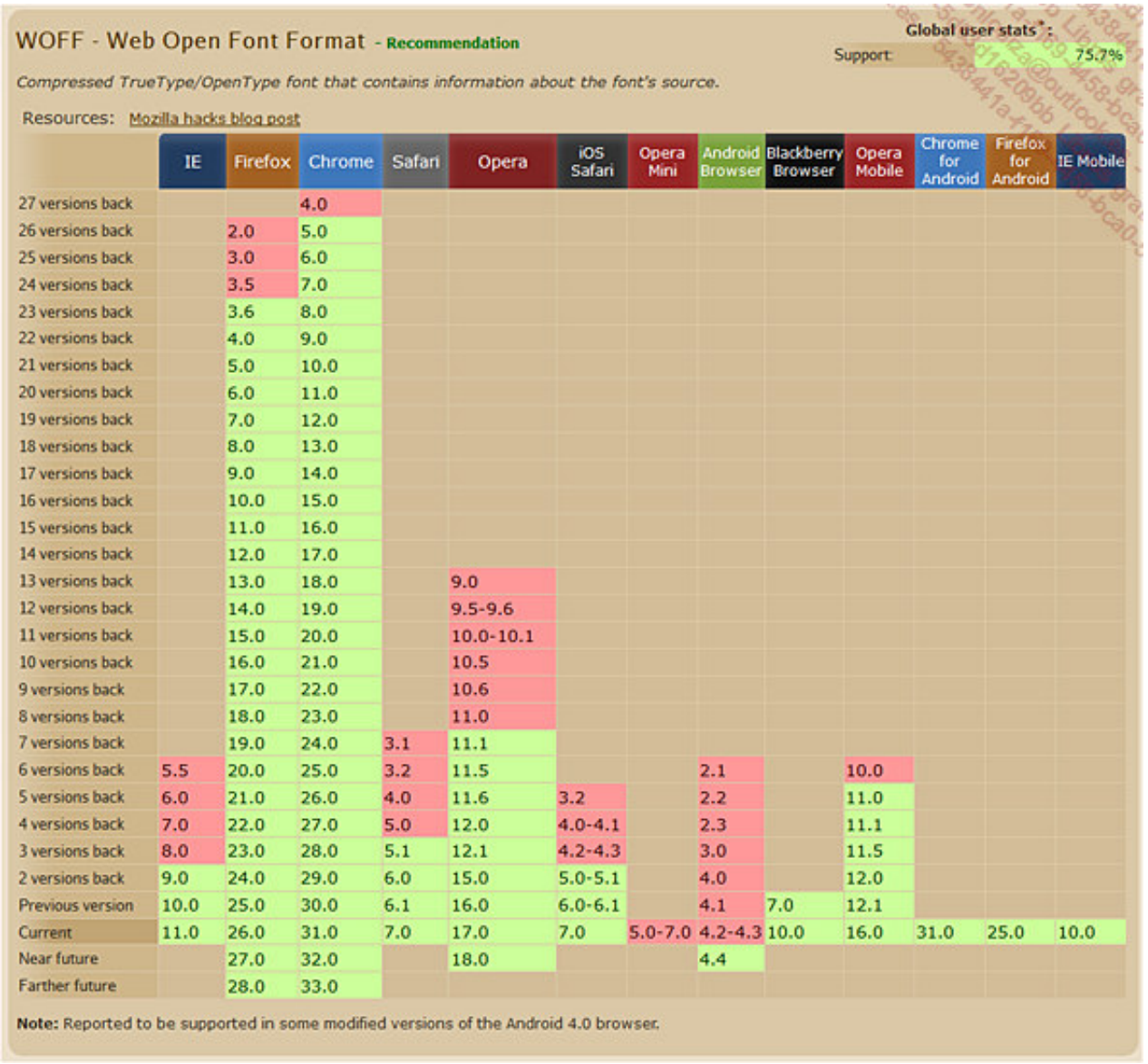
La siguiente tabla recoge las compatibilidades entre los formatos y los navegadores en su última versión (octubre de 2011).

	Safari 5	Opera 11	IE 9	Firefox 6	Chrome 14
TTF	Sí	Sí	Sí	Sí	Sí
OTF	Sí	Sí		Sí	Sí
WOFF	Sí	Sí	Sí	Sí	Sí
SVG	Sí	Sí		Sí	Sí
EOT			Sí		

Para conocer las compatibilidades de los navegadores y los diferentes formatos de fuentes, sugiero utilizar regularmente el sitio web caniuse.com.

- Compatibilidad con **TTF** y **OTF**: http://caniuse.com/ttf
- Compatibilidad con **WOFF**: http://caniuse.com/woff

Extracto de la tabla:



WOFF - Web Open Font Format - Recommendation

Global user stats*: Support: 75.7%

Compressed TrueType/OpenType font that contains information about the font's source.

Resources: Mozilla hacks blog post

	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	Opera Mobile	Chrome for Android	Firefox for Android	IE Mobile
27 versions back			4.0										
26 versions back		2.0	5.0										
25 versions back		3.0	6.0										
24 versions back		3.5	7.0										
23 versions back		3.6	8.0										
22 versions back		4.0	9.0										
21 versions back		5.0	10.0										
20 versions back		6.0	11.0										
19 versions back		7.0	12.0										
18 versions back		8.0	13.0										
17 versions back		9.0	14.0										
16 versions back		10.0	15.0										
15 versions back		11.0	16.0										
14 versions back		12.0	17.0										
13 versions back		13.0	18.0		9.0								
12 versions back		14.0	19.0		9.5-9.6								
11 versions back		15.0	20.0		10.0-10.1								
10 versions back		16.0	21.0		10.5								
9 versions back		17.0	22.0		10.6								
8 versions back		18.0	23.0		11.0								
7 versions back		19.0	24.0	3.1	11.1								
6 versions back	5.5	20.0	25.0	3.2	11.5			2.1		10.0			
5 versions back	6.0	21.0	26.0	4.0	11.6	3.2		2.2		11.0			
4 versions back	7.0	22.0	27.0	5.0	12.0	4.0-4.1		2.3		11.1			
3 versions back	8.0	23.0	28.0	5.1	12.1	4.2-4.3		3.0		11.5			
2 versions back	9.0	24.0	29.0	6.0	15.0	5.0-5.1		4.0		12.0			
Previous version	10.0	25.0	30.0	6.1	16.0	6.0-6.1		4.1	7.0	12.1			
Current	11.0	26.0	31.0	7.0	17.0	7.0	5.0-7.0	4.2-4.3	10.0	16.0	31.0	25.0	10.0
Near future		27.0	32.0		18.0			4.4					
Farther future		28.0	33.0										

Note: Reported to be supported in some modified versions of the Android 4.0 browser.

- Compatibilidad con **SVG**: http://caniuse.com/svg
- Compatibilidad con **EOT**: http://caniuse.com/eot

4. La regla @font-face

Veamos la sintaxis de la regla `@font-face`:

```
@font-face {
   font-family: "Skia";
   src: url('Skia.ttf');
}
```

Una vez que hayamos declarado la regla `@font-face`, con la propiedad `font-family`indicaremos el nombre de la fuente tipográfica que queramos importar. Luego, con la propiedad `src`indicaremos la ruta de acceso al archivo de dicha fuente.

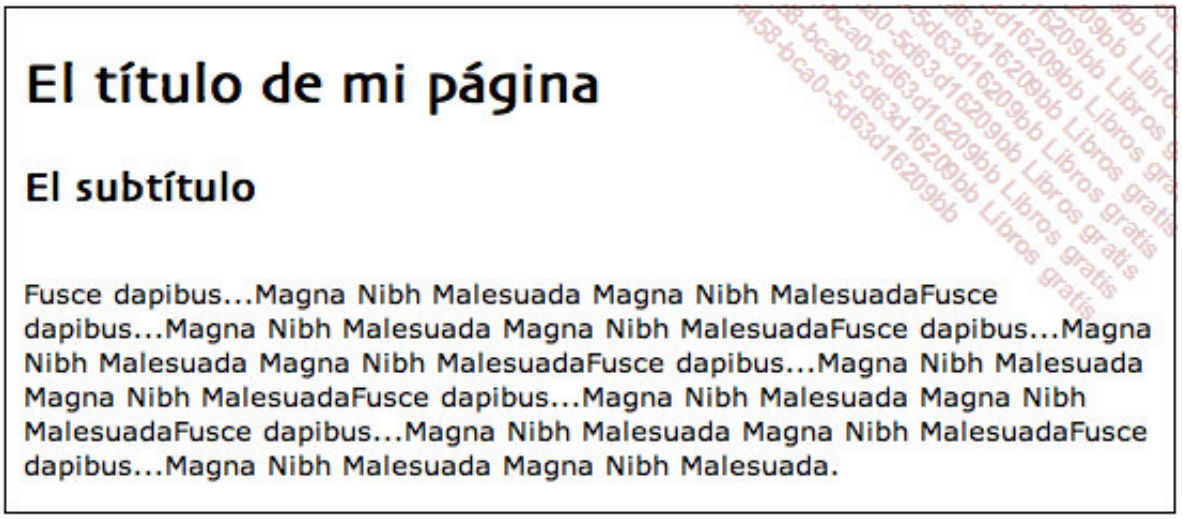
Para aplicarla, simplemente tendremos que indicar en el selector en cuestión el nombre de la fuente que deberá usarse con la propiedad `font-family`:

```
h1, h2 {
   font-family: Skia;
}
```

Este es el código HTML:

```
<h1>El título de mi página</h1>
<h2>El subtítulo</h2>
<p>Fusce dapibus...</p>
```

Y el resultado visual:



5. El nombre de las tipografías

Tenga cuidado, el nombre que se indica en `font-family` es totalmente arbitrario, no tiene por qué corresponder al nombre del archivo de la fuente tipográfica o al nombre que le haya dado el tipógrafo. Se trata del nombre que usted le haya asignado a dicha fuente. Lo que importa, es el nombre del archivo que contiene la fuente tipográfica.

Esta es una sintaxis correcta y que funciona bien:

```
@font-face {
   font-family: "Nombre de la tipografía";
   src: url('Skia.ttf');
}
h1, h2 {
   font-family: 'Nombre de la tipografía', Arial, Helvetica, sans-serif;
}
```

6. Las tipografías locales

De manera predeterminada, con la sintaxis precedente, las fuentes tipográficas se descargarán del servidor en el ordenador del usuario que visite su sitio web. Sin embargo, podemos suponer que algunos visitantes disponen de esa tipografía localmente en su ordenador. Indicaremos entonces que deberá usarse la tipografía local, cuando esta esté disponible, con `local` en `src`.

```
@font-face {
   font-family: "Skia";
   src: local("Skia"), url('Skia.ttf');
}
```

El nombre de la tipografía que se indique en "local" deberá corresponder al nombre del archivo del sistema.

Esta sintaxis se puede leer así: se debe usar una tipografía a la que el diseñador web ha llamado`Skia`, si está disponible en local se deberá usar la tipografía que se llame `Skia` en el sistema del usuario, de lo contrario, se deberá descargar la fuente que se encuentra en el archivo llamado`Skia.ttf`.

7. Indicar varias fuentes tipográficas

Podemos indicar además varios formatos de fuente (si existen) para mejorar la compatibilidad:

```
@font-face {
   font-family: "Skia";
   src: url('Skia.ttf')format("truetype"),
   url('Skia.woff')format("woff");
}
```

Claro está, también podemos indicar otras fuentes tipográficas para aumentar la compatibilidad con los navegadores antiguos.

```
h1, h2 {
   font-family: Skia, Arial, Helvetica, sans-serif;
}
```

8. Los estilos tipográficos

En la declaración de la regla `@font-face`, puede indicar determinados estilos tipográficos con la propiedad `font-weight`, `font-style` y `font-variant`. Sin embargo, los navegadores aún no los reconocen del todo bien.

```
@font-face {
   font-family: "Nombre de la tipografía";
   src: url('Skia.ttf');
   font-weight: bold;
}
```

9. Un generador de fuentes tipográficas con @font-face

El servicio en línea **CSS 3.0 Maker** (http://www.css3maker.com/index.html) propone varios generadores CSS3, entre ellos, un generador de fuentes tipográficas con la regla `@font-face`:http://www.css3maker.com/font-face.html

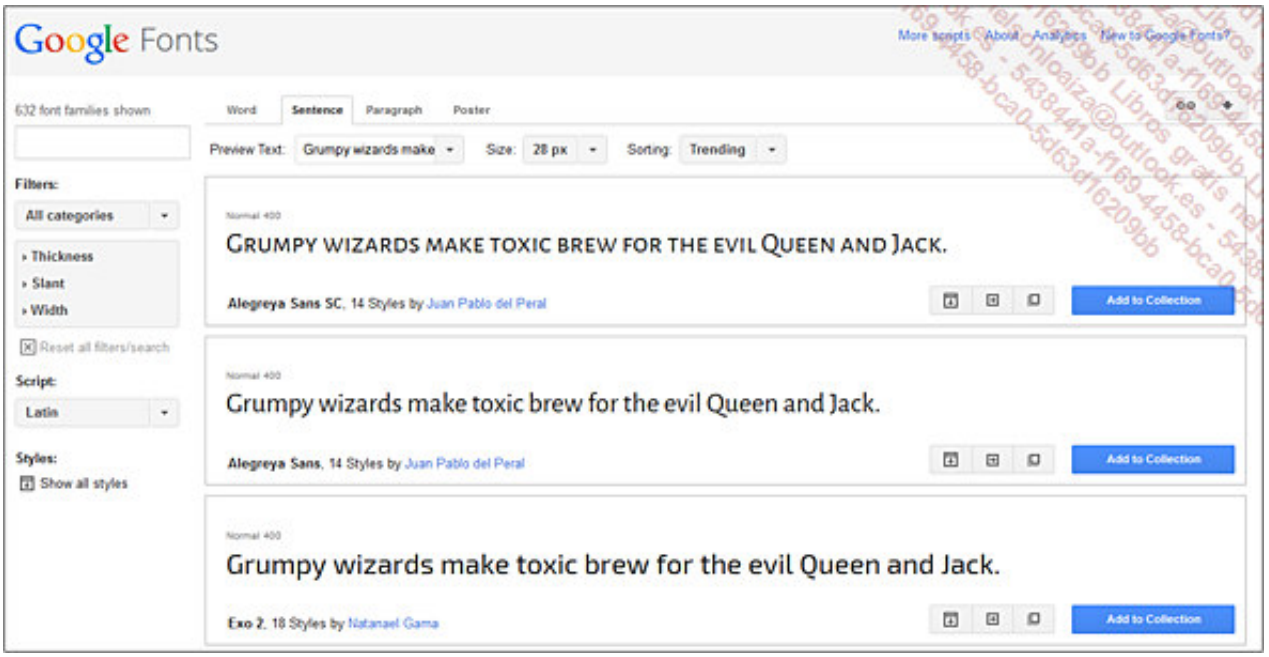
En esta interfaz de color oscuro (y poco práctica), en el recuadro **CSS3 Styles**, seleccione una tipografía, el tamaño, el estilo y el ancho. En el recuadro **CSS3 Preview Area** podrá visualizar el

resultado de su selección. En el recuadro **CSS3 Codeview** podrá copiar el código generado, o bien, puede hacer clic en el botón **Download**.

10. Las tipografías web en línea

Diferentes servicios en línea ofrecen la posibilidad de usar tipografías para la Web. El servicio de Google, **Google Fonts**, es uno de los más populares. Esta es su URL: http://www.google.com/fonts

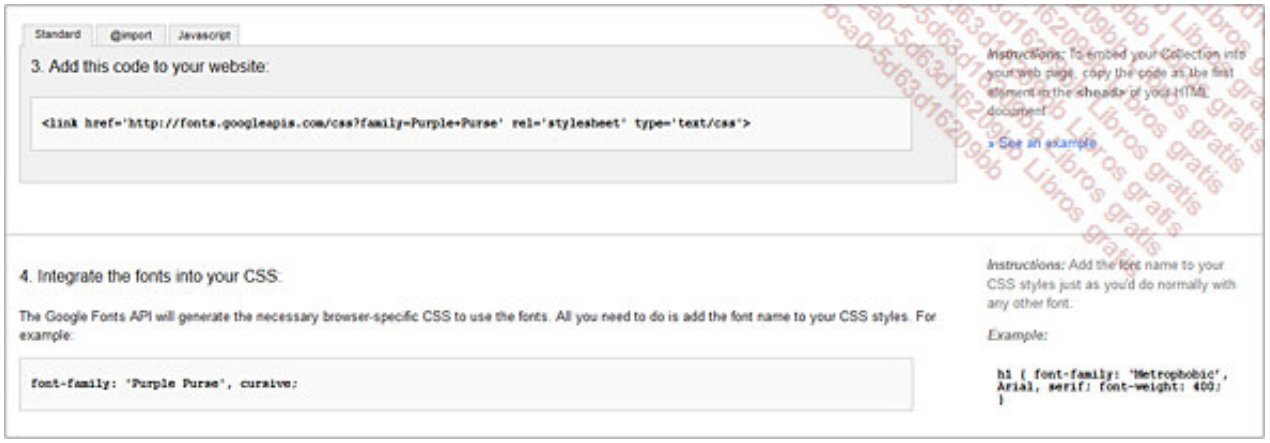
Dispone de más de 630 tipografías.



En la lista de tipografías, seleccione la que usted prefiera.

En el recuadro, puede hacer clic en el botón **Pop out**  para obtener más información sobre esa tipografía.

En el recuadro precedente, haga clic en el botón **Quick-use** . En los recuadros **3**. y **4**. encontrará las instrucciones para usar esa tipografía en sus hojas de estilo CSS.



Este sería un ejemplo de vínculo CSS:

```
<link href='http://fonts.googleapis.com/css?family=Julee' rel='stylesheet'
type='text/css'>
```

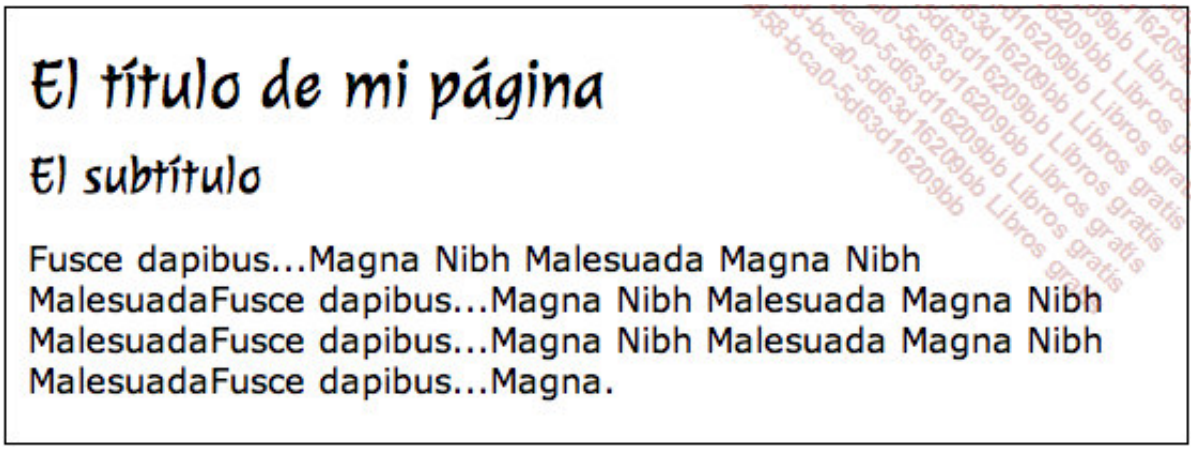
Y la aplicación en la hoja de estilo:

```
h1, h2 {
   font-family: 'Julee', cursive;
}
```

Veamos el código HTML de la página:

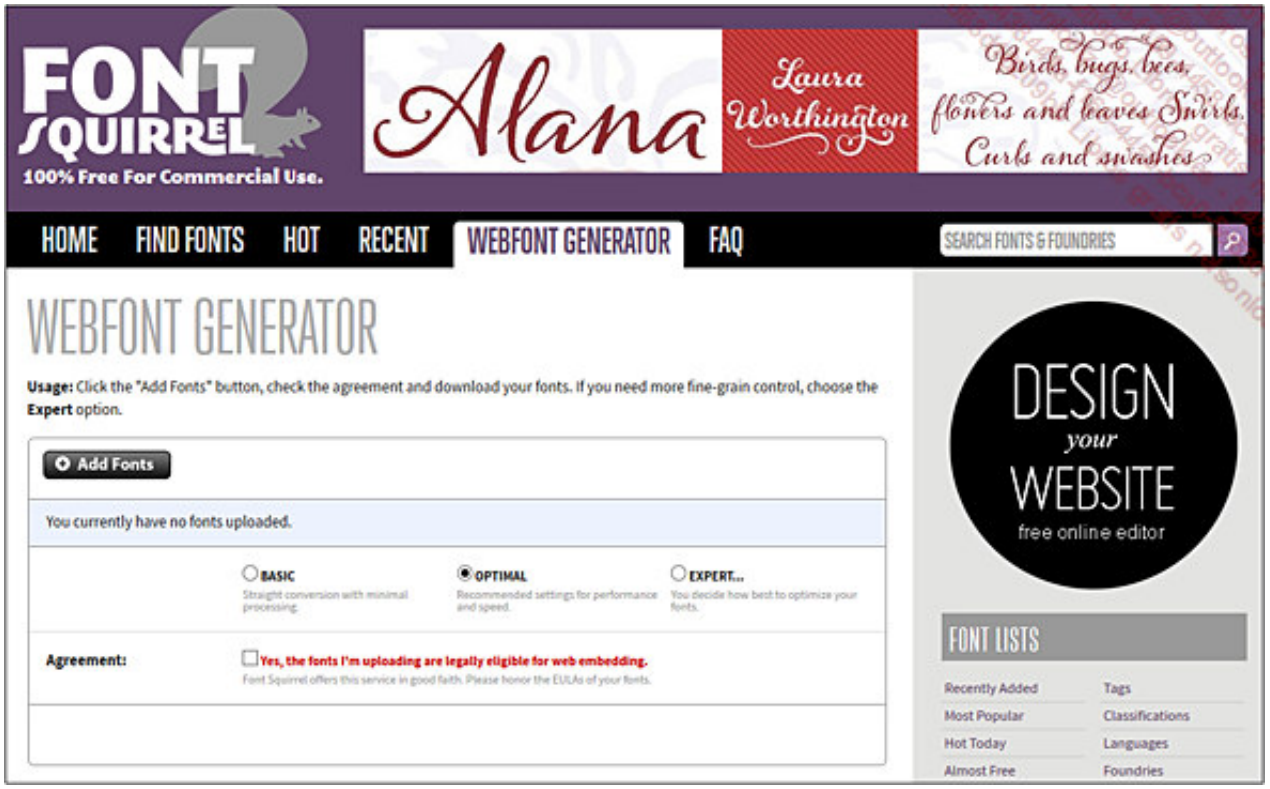
```
<h1>El título de mi página</h1>
<h2>El subtítulo</h2>
<p>Fusce dapibus...</p>
```

Este es el resultado visual:



11. Convertir las tipografías

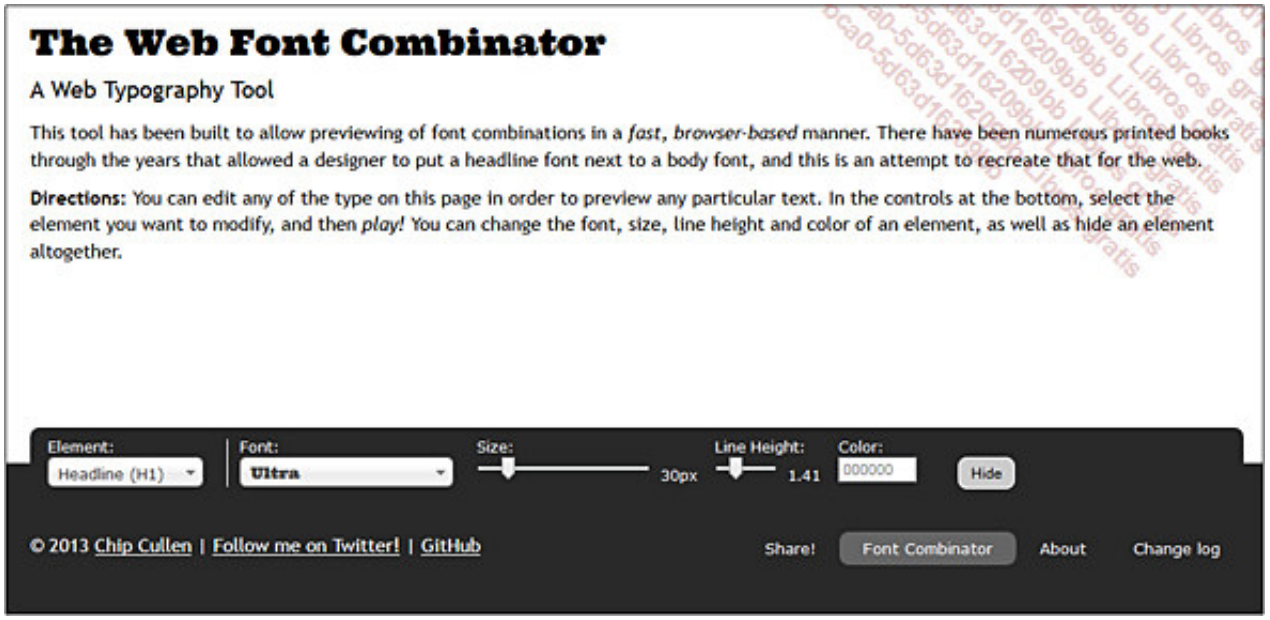
El sitio web **Font Squirrel** (http://www.fontsquirrel.com/) es uno de los sitios web más populares para convertir las tipografías a diversos formatos (http://www.fontsquirrel.com/tools/webfont-generator).



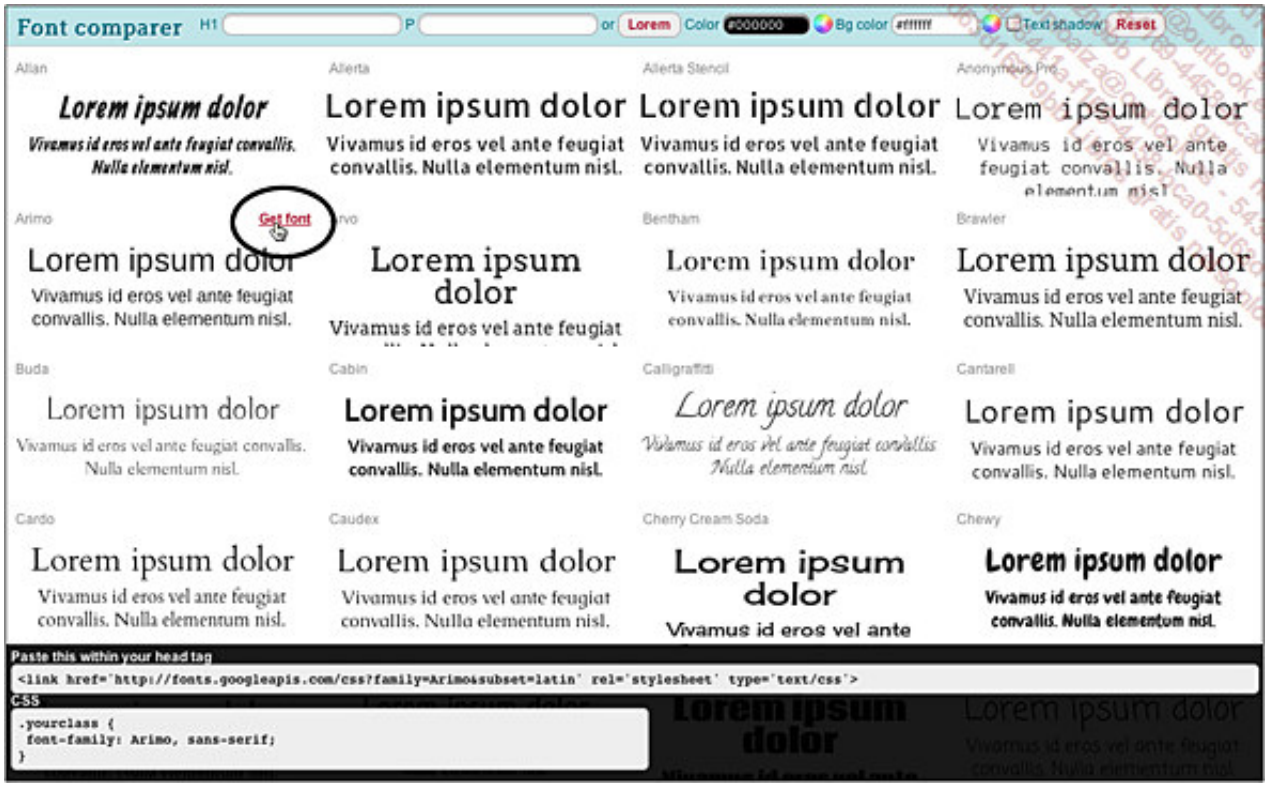
12. Seleccionar las tipografías

A la hora de diseñar la interfaz de un sitio web, el sitio web **Font Combinator** (http://font-combinator.com/) le permite probar en línea tres fuentes tipográficas de Google: una para los títulos, una para los subtítulos y otra para el texto normal. Resulta muy práctico para examinar cómo se visualizarán las tres fuentes tipográficas que haya seleccionado.

Seleccione las tipografías y el tamaño en la parte derecha de la interfaz.



Otro sitio web interesante es **Font comparer** (http://www.fontcomparer.com/). En este sitio podrá comparar un gran número de tipografías de Google y luego recuperar el vínculo de la tipografía que desee. Para ello, pase el cursor por encima del nombre de una tipografía y haga clic en el vínculo **Get font**.


Font comparer
H1
P
or
Lorem
Color
#000000
Bg color
#ffffff
Text shadow
Reset
Allan
Lorem ipsum dolor
Vivamus id eros vel ante feugiat convallis. Nulla elementum nisl.
Allerta
Lorem ipsum dolor
Vivamus id eros vel ante feugiat convallis. Nulla elementum nisl.
Allerta Stencil
Lorem ipsum dolor
Vivamus id eros vel ante feugiat convallis. Nulla elementum nisl.
Anonymous Pro
Lorem ipsum dolor
Vivamus id eros vel ante feugiat convallis. Nulla elementum nisl
Arimo
Get font
Lorem ipsum dolor
Vivamus id eros vel ante feugiat convallis. Nulla elementum nisl.
Arvo
Lorem ipsum dolor
Vivamus id eros vel ante feugiat
Bentham
Lorem ipsum dolor
Vivamus id eros vel ante feugiat convallis. Nulla elementum nisl.
Brawler
Lorem ipsum dolor
Vivamus id eros vel ante feugiat convallis. Nulla elementum nisl.
Buda
Lorem ipsum dolor
Vivamus id eros vel ante feugiat convallis. Nulla elementum nisl.
Cabin
Lorem ipsum dolor
Vivamus id eros vel ante feugiat convallis. Nulla elementum nisl.
Calligraffiti
Lorem ipsum dolor
Vivamus id eros vel ante feugiat convallis Nulla elementum nisl
Cantarell
Lorem ipsum dolor
Vivamus id eros vel ante feugiat convallis. Nulla elementum nisl.
Cardo
Lorem ipsum dolor
Vivamus id eros vel ante feugiat convallis. Nulla elementum nisl.
Caudex
Lorem ipsum dolor
Vivamus id eros vel ante feugiat convallis. Nulla elementum nisl.
Cherry Cream Soda
Lorem ipsum dolor
Vivamus id eros vel ante
Chewy
Lorem ipsum dolor
Vivamus id eros vel ante feugiat convallis. Nulla elementum nisl.
Paste this within your head tag
<link href='http://fonts.googleapis.com/css?family=Arimo&subset=latin' rel='stylesheet' type='text/css'>
CSS
.yourclass {
font-family: Arimo, sans-serif;
}

El tamaño de las tipografías

1. Las diferentes medidas

Como ya sabe, existen dos "formatos" para definir el tamaño de los caracteres:

- las **medidas absolutas**: `cm`, `mm`, `in`, `pt`... No es aconsejable usar esas medidas, ya que no son conformes a las recomendaciones de accesibilidad para los sitios web. Estas generan problemas de redimensión del texto en las páginas web para las personas con discapacidad visual.
- las **medidas relativas**: `em` y `%`. Es preferible usar estas unidades relativas, ya que se pueden redimensionar sin problemas. Para todos los navegadores, los tamaños son relativos al tamaño por defecto de los párrafos `p`: 1em = 100% = 16px.

2. El problema de la cascada

Las hojas de estilo se aplican en cascada. Esto implica que los tamaños también se aplican en cascada.

Veamos un ejemplo de una hoja de estilo:

```
body {
   font-size: 1em;}
li {
   font-size: .8em;
}
```

En esta sencilla página tenemos dos listas imbricadas:

```
<h1>Título de la página</h1>
<ul>
    <li>Primero</li>
    <li>
       <ul>
       <li>Uno</li>
       <li>Dos</li>
       <li>Tres</li>
       </ul>
    </li>
    <li>Segundo</li>
    <li>Tercero</li>
</ul>
```

Este es el resultado visual:

¿Cómo se calcula el tamaño de los caracteres? El tamaño es de `1em` a nivel del `body`, lo que determina que el tamaño mostrado y calculado sea de 16 píxeles para el texto normal. El `h1` se calcula proporcionalmente a 2 em, es decir, 32 píxeles según el cálculo. Los elementos `li` de la primera lista son de `0.8em`, es decir, 16x0.8=12.8 píxeles según el cálculo, así que se visualizarán a 13 píxeles. Los `li` de la segunda lista imbricada se calculan con la cascada a 16x0.8x0.8=10,24 píxeles, lo que quiere decir que se visualizarán a 10 píxeles. El tamaño en em se calcula en relación al elemento padre.

3. La nueva unidad rem

La nueva unidad **rem** del CSS3 realiza el cálculo en relación a la raíz de la página, en función del elemento `html`. La unidad rem significa **root em**. Podemos definir el tamaño relativo que deseemos para el elemento `html` con la unidad `em`, y luego todos los demás tamaños con la unidad `rem`. ¡Así no tendremos problemas con las cascadas!

Si volvemos al ejemplo anterior, ahora deberíamos indicar en la hoja de estilo:

```
html {
     font-size: 1em;
}
body {
     font-size: .8rem;
}
li {
     font-size: .8rem;
}
```

Ya no tenemos ningún efecto de cascada en el cálculo del tamaño de los elementos. Todos los `li`tienen el mismo tamaño de carácter.

Título de la página

- Primero
-
    - Uno
    - Dos
    - Tres
- Segundo
- Tercero

Textos con sombra

1. Los valores de la propiedad

La nueva propiedad CSS3 `text-shadow` permite aplicar una sombra a un texto. Esta propiedad puede usar diversos valores:

- el primer valor es el desplazamiento horizontal de la sombra.
- el segundo valor es el desplazamiento vertical de la sombra.
- el tercer valor es el tamaño del difuminado de la sombra.
- el cuarto valor es el color de la sombra.
- el quinto valor es la dirección de la sombra: `inset`, hacia el interior (por defecto, la sombra será exterior).

Veamos un ejemplo sencillo:

```
h1 {
    text-shadow: 8px 8px 5px rgb(100,50,200);
}
```

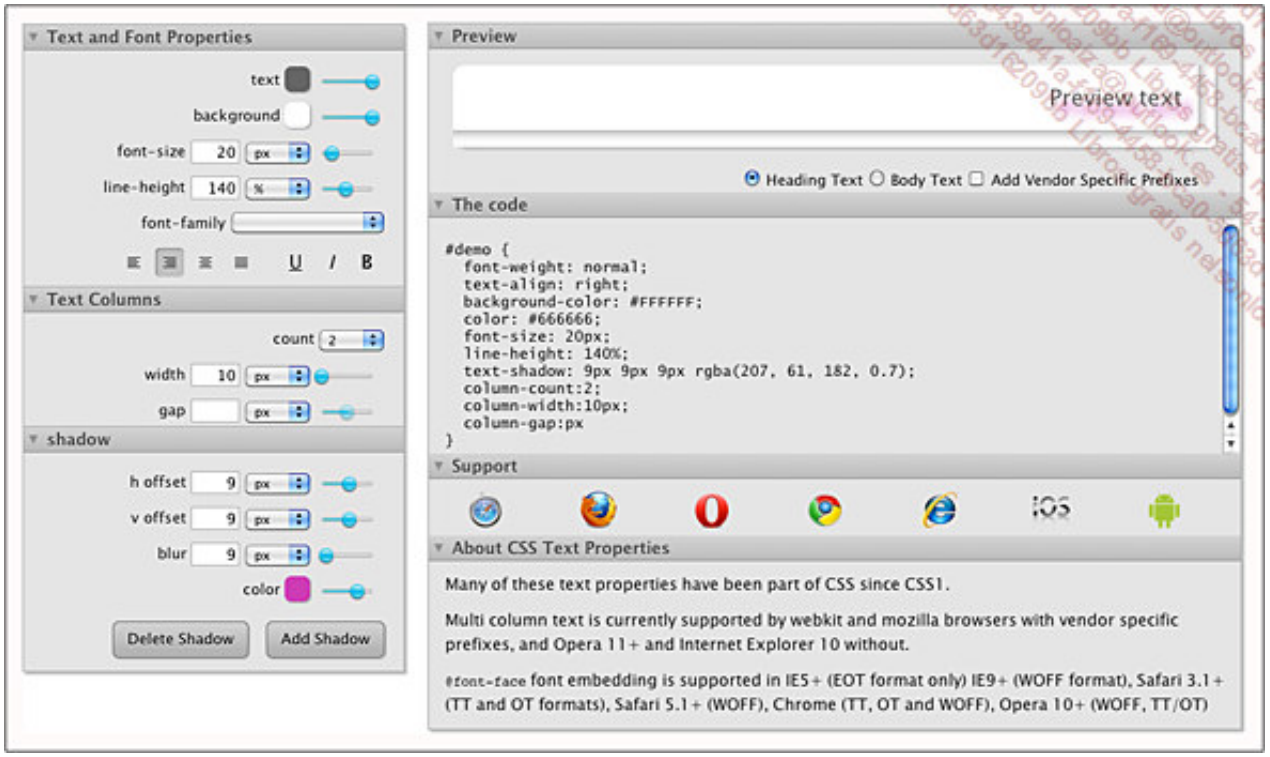
- desplazamiento horizontal de 8 píxeles,
- desplazamiento vertical de 8 píxeles,
- difuminado de 5 píxeles,
- color violeta.

Este es el resultado visual:

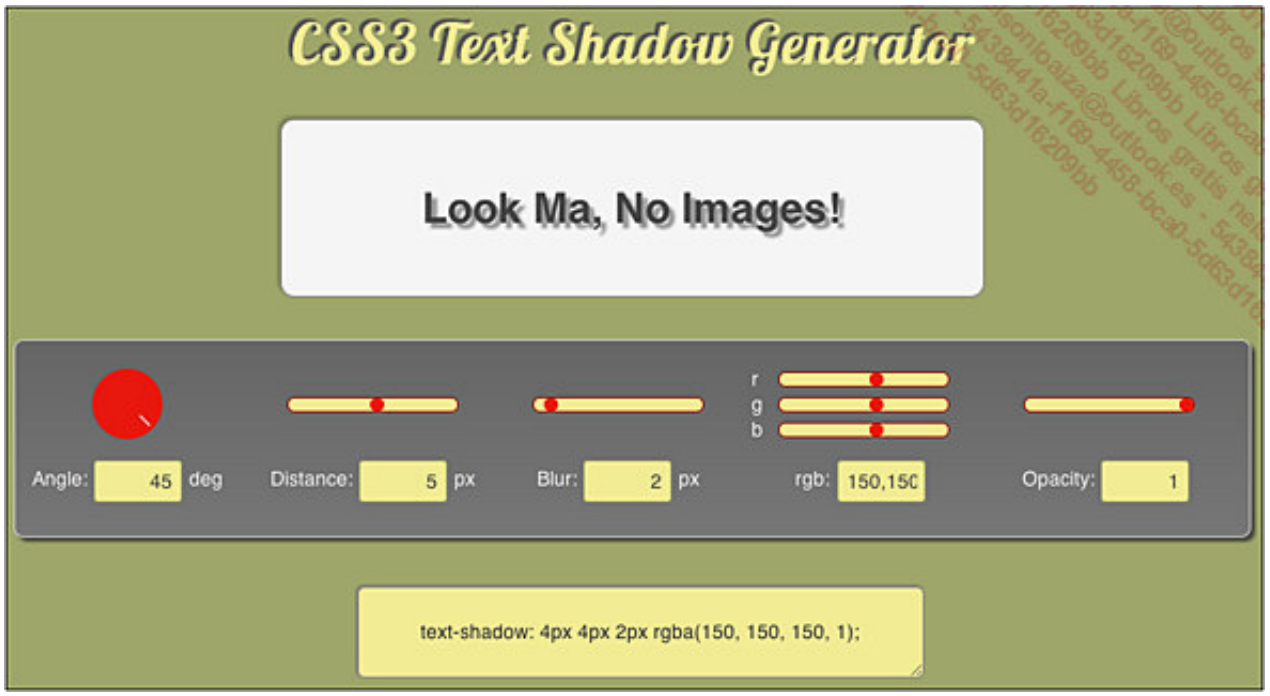
2. Las herramientas en línea

Una vez más, disponemos de multitud de soluciones en línea para generar texto con sombras.

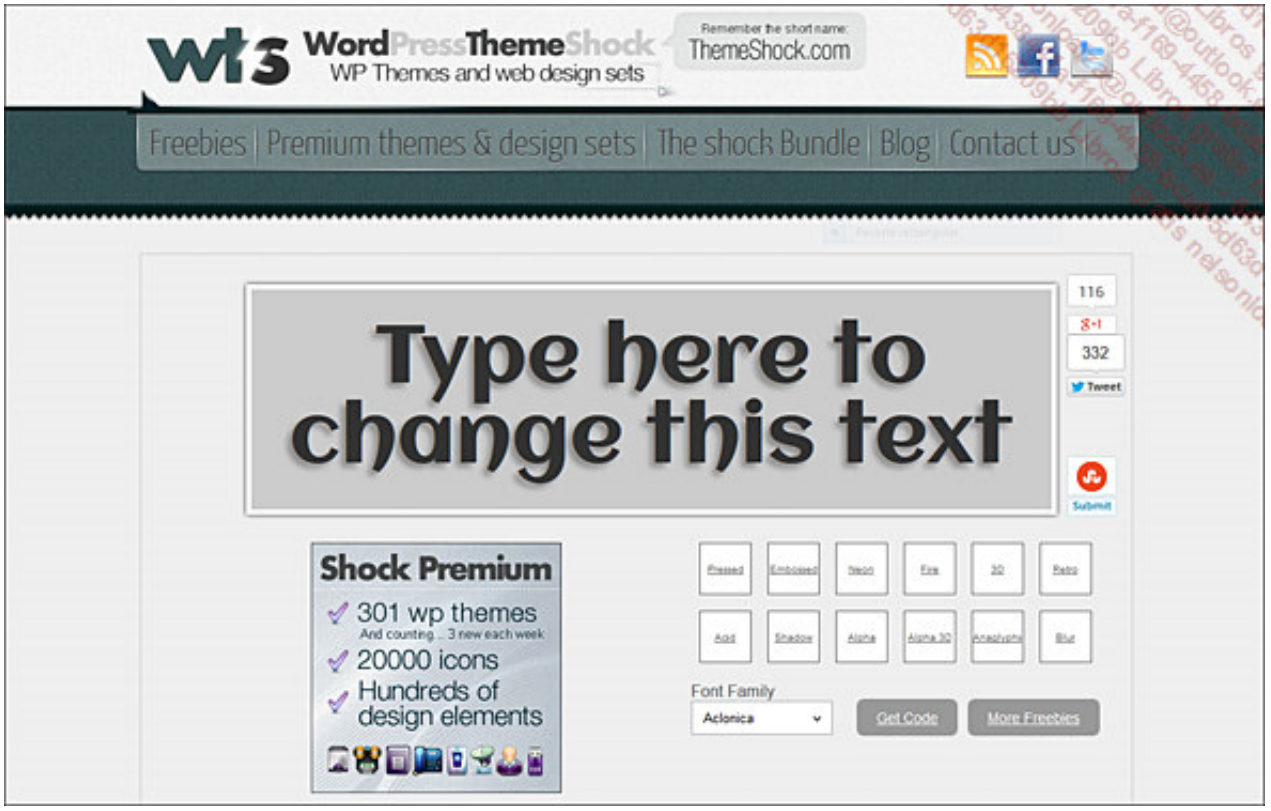
El sitio web **WestCIV** (http://www.westciv.com/) pone a su disposición completas herramientas para generar estilos CSS3. La herramienta **Text Properties** (http://www.westciv.com/tools/text-properties/index.html) presenta una completa interfaz en la que podrá dar formato a su texto con las propiedades CSS3. En la zona **Text and Font Properties**, encontrará las propiedades clásicas que permiten cambiar el formato del texto. En la zona **Text Columns**, podrá organizar su caja en columnas y, en la zona **shadow** , podrá aplicar una sombra a su texto.



El sitio web **CSS3Gen** (http://css3gen.com/text-shadow/) pone a su disposición una sencilla interfaz donde podrá generar el código CSS3 que necesite.



El sitio web **WordPressThemeShock** también le propone un generador de texto sombreado (http://www.wordpressthemeshock.com/css-text-shadow/).



3. Algunos ejemplos de diseño

Estos son algunos ejemplos sacados del sitio web http://webexpedition18.com

Un efecto retro mediante la aplicación de dos sombras:

```
@font-face {
    font-family: "Hobo";
    src: url("HoboStd.otf");
}
h1 {
    border: solid 1px #ccc;
    padding: 5px;
    width: 300px;
    background-color: #fff;
    font-family: "Hobo";
    color: rgb(158,17,10);
    text-shadow: 1px 1px 0px white, 2px 2px 0px grey;
    text-align: center;
}
```

El resultado visual:

Un efecto 3D con multitud de sombras desplazadas para simular el efecto 3D:

Los estilos CSS:

```
@font-face {
    font-family: "Hobo";
```

```
    src: url("HoboStd.otf");
}
h1 {
    background-color: #fce;
    width: 300px;
    padding: 5px;
    font-family: "Hobo";
    color: rgb(158,17,10);
    text-shadow: 0 1px 0 #ccc,
                 0 2px 0 #c9c9c9,
                 0 3px 0 #bbb,
                 0 4px 0 #b9b9b9,
                 0 5px 0 #aaa,
                 0 6px 1px rgba(0,0,0,.1),
                 0 0 5px rgba(0,0,0,.1),
                 0 1px 3px rgba(0,0,0,.3),
                 0 3px 5px rgba(0,0,0,.2),
                 0 5px 10px rgba(0,0,0,.25),
                 0 10px rgba(0,0,0,.2),
                 0 20px 20px rgba(0,0,0,.15);
    text-align: center;
}
```

El resultado obtenido:

Un efecto de desplazamiento negativo:

```
@font-face {
    font-family: "Hobo";
    src: url("HoboStd.otf");
}
h1 {
    border: solid 1px #ccc;
    padding: 5px;
    width: 300px;
    background-color: #ddd;
    font-family: "Hobo";
    color: rgb(158,17,10);
    text-shadow:-2px -1px 0 #fff;
    text-align: center;
}
```

El resultado visual:

Una sombra clara sobre un fondo oscuro:

```
@font-face {
    font-family: "Hobo";
```

```
    src: url("HoboStd.otf");
}
h1 {
    border: solid 1px #ccc;
    padding: 5px;
    width: 300px;
    background-color: #474747;
    font-family: "Hobo";
    color:#222;
    text-shadow: 0px 1px 2px #eee;
    text-align: center;
}
```

El resultado obtenido:

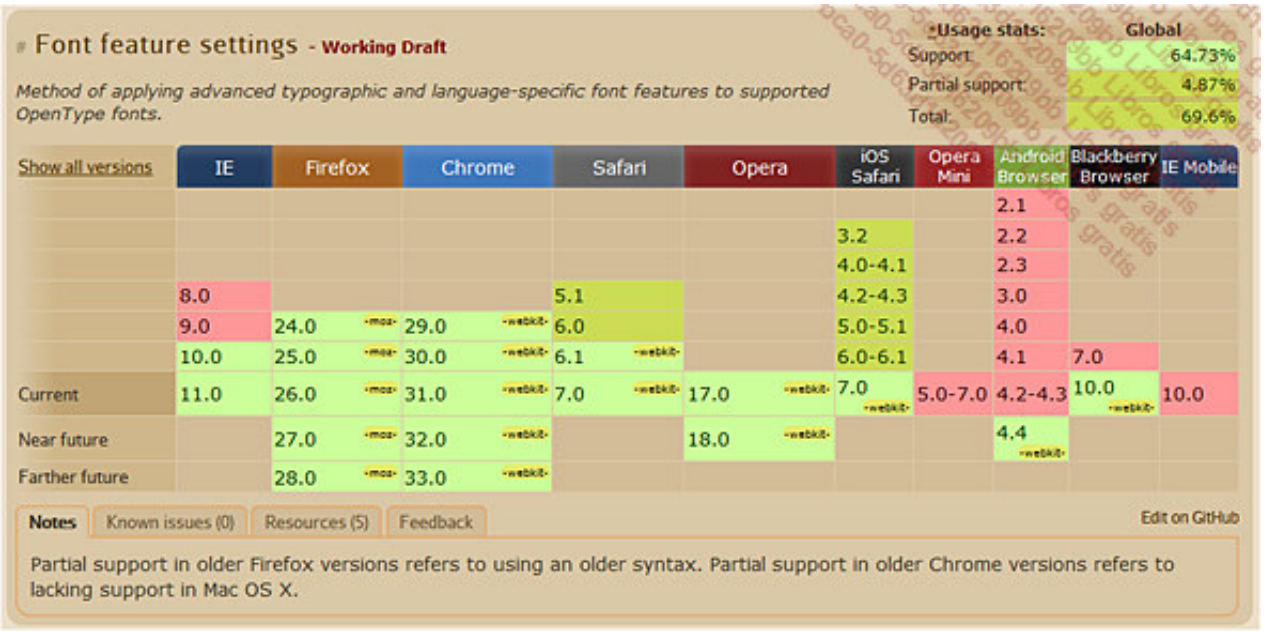
La tipografía avanzada

1. Situación actual

Los estilos CSS3 relacionados con las fuentes tipográficas (http://www.w3.org/TR/css3-fonts/, en**Last Call Working Draft** el 11 de julio de 2013) no están aún lo suficientemente reconocidos, en cuanto a tipografía avanzada se refiere.

Para usar esas propiedades avanzadas, deberá usar los prefijos de los navegadores y hacer innumerables tests.

Esta es la tabla de compatibilidad de acuerdo con canieuse.com:



Font feature settings - Working Draft

Method of applying advanced typographic and language-specific font features to supported OpenType fonts.

*Usage stats: Global
Support: 64.73%
Partial support: 4.87%
Total: 69.6%

Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	IE Mobile
								2.1		
						3.2		2.2		
						4.0-4.1		2.3		
	8.0			5.1		4.2-4.3		3.0		
	9.0	24.0 -moz-	29.0 -webkit-	6.0		5.0-5.1		4.0		
	10.0	25.0 -moz-	30.0 -webkit-	6.1 -webkit-		6.0-6.1		4.1	7.0	
Current	11.0	26.0 -moz-	31.0 -webkit-	7.0 -webkit-	17.0 -webkit-	7.0 -webkit-	5.0-7.0	4.2-4.3	10.0 -webkit-	10.0
Near future		27.0 -moz-	32.0 -webkit-		18.0 -webkit-			4.4 -webkit-		
Farther future		28.0 -moz-	33.0 -webkit-							

Notes | Known issues (0) | Resources (5) | Feedback | Edit on GitHub

Partial support in older Firefox versions refers to using an older syntax. Partial support in older Chrome versions refers to lacking support in Mac OS X.

2. El grosor de los caracteres

Como ya sabe, dispone de los valores `normal`, `bold`, `bolder` y `lighter` para definir el grosor de los caracteres. Desde el CSS2 el W3C ya proponía una escala relativa para harmonizar los términos empleados, que pueden ser muy diferentes de un tipógrafo a otro. Se trata de una escala que va de 100 a 900.

- `100` = Thin
- `200` = Extra Light (Ultra Light)
- `300` = Light
- `400` = Normal
- `500` = Medium
- `600` = Semi Bold (Demi Bold)
- `700` = Bold
- `800` = Extra Bold (Ultra Bold)
- `900` = Black (Heavy)

3. El tamaño de los caracteres

La propiedad `font-stretch` permite definir la caja, la extensión de los caracteres, para obtener caracteres más estrechos (`condensed`) o más anchos (`expanded`). Los valores posibles son:`Ultra Condensed`, `Extra Condensed`, `Condensed`, `Semi Condensed`, `Normal`, `Semi Expanded`, `Expanded`, `Extra Expanded`, `Ultra Expanded`.

4. Ajustar el tamaño de los caracteres

El tamaño de los caracteres se ajustará en función de la altura de las astas ascendentes y descendentes. Visualmente, determinadas tipografías presentan diferentes alturas para las minúsculas. La fuente Verdana (creada específicamente para mejorar la lectura en Internet) tiene minúsculas con una altura más importante que la Arial o la Helvética.

La propiedad `font-size-adjust` permite ajustar el tamaño de los caracteres en función de la altura de las minúsculas, y no de la altura clásica de los caracteres.

El valor que se le atribuya a esta propiedad será un multiplicador del tamaño habitual.

Estos son los estilos CSS utilizados en el ejemplo, sin ajustar:

```
body {
    font-family: verdana;
}
@font-face {
    font-family: "Arial";
    src: "Arial.ttf";
}
.titulo{
    font-family: Arial;
}
```

Este es el código HTML:

```
<h1 class="titulo">Mi título en Arial</h1>
<h1>Mi título en Verdana</h1>
```

Así se visualizará en Firefox:

Podemos ver claramente la diferencia de talla entre ambas tipografías, Arial y Verdana, a pesar de que tengan el mismo tamaño de carácter.

Añadamos ahora la propiedad `font-size-adjust` para ajustarlas y atenuar esa diferencia de altura de las minúsculas:

```
.titulo{
    font-family: Arial;
    font-size-adjust: .56;
}
```

Así se visualizará en Firefox:

Usted deberá encontrar el valor adecuado para el ajuste.

5. Otras propiedades

En el futuro también dispondremos de las propiedades para el espacio entre los caracteres (`font-kerning`), la unión de caracteres (`font-variant-ligatures`), los caracteres numéricos especiales, como las fracciones (`font-variant-numeric`)...

Funcionalidad

Las páginas en HTML 4 que contienen formularios requieren que se validen los valores recuperados, tanto del lado del servidor, como del lado del cliente, mediante JavaScript. Además, los tipos de contenido suelen ser muy generalistas y hay pocas interacciones con los usuarios. HTML5 permite dar un paso importante para mejorar esos defectos.

Los métodos de envío

HTML 4 permitía enviar los datos de un formulario mediante dos métodos: `get` y `post`. HTML5 introduce dos nuevos tipos de envío: `put` y `delete`. Sin embargo, la implementación de estos dos nuevos métodos presenta algunos problemas y por el momento no se ha llegado a ninguna conclusión. Conviene seguir de cerca su evolución, para ver si finalmente se validan o no.

El elemento <form>

En HTML 4, el elemento `<form>`, que permite crear formularios, debe contener todos los elementos de dicho formulario (campos de introducción de datos, botones, casillas...). En HTML5 es posible "sacar" algunos elementos del formulario.

Veamos un ejemplo de un elemento `<textarea>` situado fuera del formulario:

```
<form id="registro">
    <input type="text">
    ...
</form>
<textarea form="registro"></textarea>
```

El primer campo de introducción de datos (`<input>`) se encuentra dentro del formulario (`<form>`), que dispone de un código de identificación `id` único. El campo de texto (`<textarea>`) se encuentra fuera del formulario, pero está vinculado al mismo mediante el atributo `form`, que presenta como valor el `id` de dicho formulario.

Los nuevos campos

1. La visualización de los nuevos campos

Como es habitual, la especificación del HTML5 no define cómo deberán mostrarse los elementos de los formularios. Cada navegador usará sus propios elementos de interfaz. De este modo, habrá una gran diferencia entre el navegador de un ordenador, el de un smartphone y el de una tableta gráfica.

Recuerde que, de manera predeterminada, los campos de introducción de datos son de tipo`<input type="text">`, lo que quiere decir que si un navegador no reconoce un nuevo tipo de campo, lo presentará como un campo de texto "normal".

2. Los campos para las direcciones de e-mail

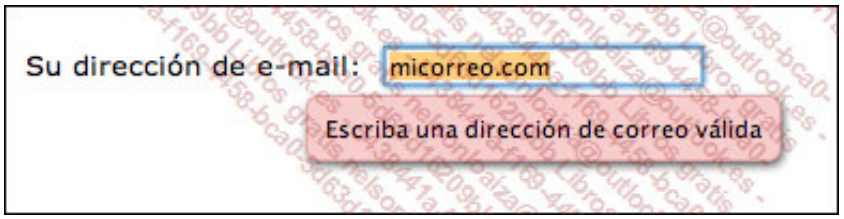
El nuevo campo de introducción de datos de tipo `email` permite especificar que el contenido insertado deberá ser una dirección de e-mail, es decir, que deberá incluir el carácter `@`. A continuación serán los navegadores quienes comprobarán de forma nativa si la arroba ha sido insertada o no.

Esta es la sintaxis: `<input type="email">`

Otra ventaja innegable, cuando se use con un smartphone que reconozca este tipo de campo: el teclado se adaptará automáticamente para la inserción de la dirección de e-mail. Ejemplo con un iPhone:

Sepa, además, que será el navegador quien decida con total libertad cómo va a reaccionar cuando el valor introducido no sea una dirección de e-mail válida.

Este es el mensaje de error en el navegador Opera:



3. El campo para los números de teléfono

El campo de inserción de datos de tipo `tel` ha sido creado para recuperar números de teléfono.

Esta sería la sintaxis: `<input type="tel">`

No se aplicará ninguna restricción, ya que los números de teléfono son muy diferentes de un país a otro, e incluso pueden contener caracteres que no sean numéricos. Una vez más, los smartphones podrán adaptar el teclado.

4. El campo para las URL

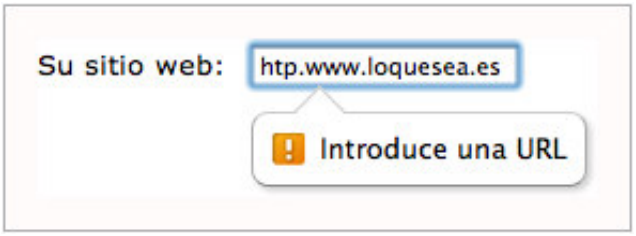
El campo para la introducción de datos de tipo `url` permite recuperar una URL. La comprobación de

dicha URL para ver si es válida dependerá totalmente del navegador.

Esta sería la sintaxis: `<input type="url">`

La última versión del navegador Opera añade automáticamente "http" cuando se introduce una URL que comience por "www". Una vez más, los smartphones podrán adaptar el teclado para facilitar la inserción de la URL.

Este es el mensaje de aviso que muestra el navegador Chrome si la URL facilitada no es válida:

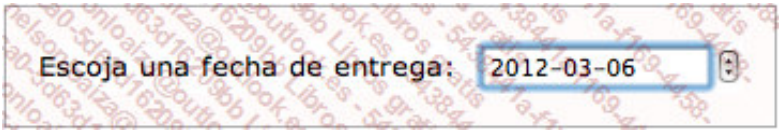


5. Los campos para las fechas y las horas

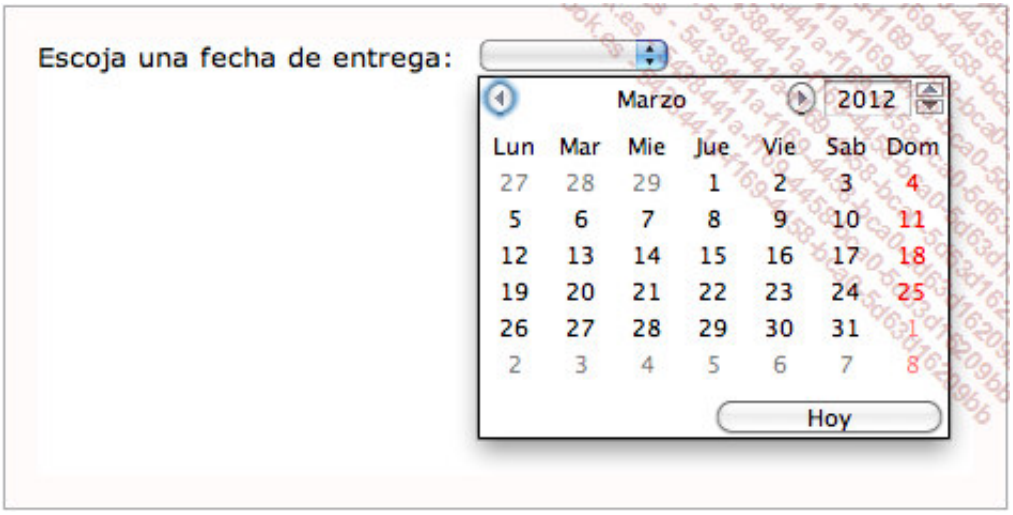
El campo de tipo `date` permite insertar una fecha. Pero, en lugar de dejar que sean los propios usuarios quienes inserten las fechas, con los riesgos de error que eso conlleva, los navegadores propondrán de forma nativa una interfaz de selección de fechas que sea fácil de usar.

Esta sería la sintaxis: `<input type="date">`

Esta es la interfaz minimalista que proponen Chrome y Safari, con el formato AAAA-MM-DD:



Opera propone una interfaz más ergonómica:



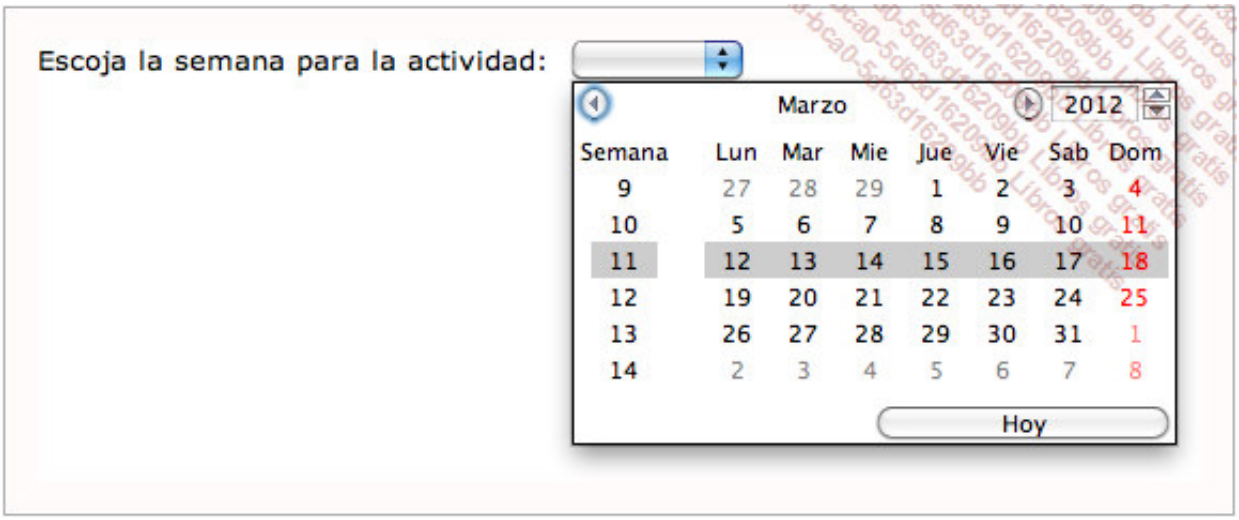
Las fechas también presentan el formato AAAA-MM-DD:

Existen otros tipos para los campos de fechas:

- el día y la hora con la diferencia horaria: `<input type="datetime">`
- el día y la hora sin la diferencia horaria: `<input type="datetime-local">`
- la fecha solamente: `<input type="date">`
- la hora solamente: `<input type="time">`
- las semanas solamente: `<input type="week">`
- los meses solamente: `<input type="month">`

Este es un ejemplo de inserción de la hora con Opera:

Este es un ejemplo de inserción de la semana con Opera:



Y una vez que hemos elegido la semana:

6. El campo para los valores numéricos

Con el campo de inserción de datos de tipo `number` solamente se pueden introducir valores numéricos. La validación de este tipo de campo se deja una vez más a la libre elección de los navegadores.

Esta sería la sintaxis: `<input type="number">`

Este tipo de campo admite varios atributos:

- `min`: el valor mínimo aceptado.
- `max`: el valor máximo aceptado.
- `step`: el valor del incremento que deberá aplicarse en la interfaz.
- `value`: el valor que se haya facilitado.

Veamos un ejemplo de sintaxis en el que podemos elegir un valor numérico, de 18 como mínimo, hasta un máximo de 100, mediante intervalos de 2 en 2 y con un valor inicial de 48:

```
<input type="number" min="18" max="100" step="2" value="48" id="edad" />
```

Esta es la interfaz que le propondrá Opera:

7. Las barras de selección con cursor

El campo de introducción de datos `range`, que en realidad no permite introducir datos, mostrará una barra de selección con un cursor en la que podremos elegir un valor.

Esta sería la sintaxis: `<input type="range">`

Este tipo de campo admite los mismos atributos que `number`.

La visualización de los valores seleccionados se realiza con JavaScript:

```
<p>
<label for="edad">Indique su edad: </label>
<input type="range" min="18" max="100" step="2" value="48" id="edad2"
oninput="document.getElementById('valor').textContent=value">
<output id="valor">0</output>
</p>
```

Así se mostrará la barra de selección en Safari:

8. Los campos de búsqueda

El tipo `search` permite insertar un campo de introducción de datos específico para las búsquedas. Su visualización dependerá de los parámetros de la interfaz del navegador que esté utilizando el visitante, ¡así que puede variar muchísimo!

Esta sería la sintaxis: `<input type="search">`

Así se visualizará en Safari :

9. El campo de introducción de datos con sugerencias

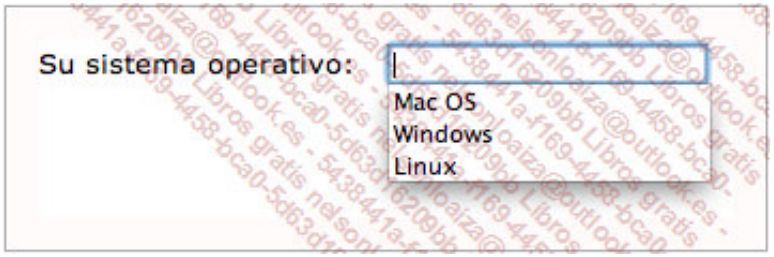
Este campo de tipo `datalist` permite disponer de un campo de inserción libre y, al mismo tiempo, de las funcionalidades de un menú desplegable. De este modo el usuario podrá insertar un valor personalizado o bien elegir un valor entre las sugerencias propuestas.

El elemento `datalist` utiliza los elementos habituales `option` para las sugerencias. Tendremos que usar simultáneamente un campo de tipo `text` y un elemento `datalist`. La relación se establece con el valor del atributo `list` de `input` y el `id` de `datalist`.

Veamos un ejemplo de aplicación:

```
<input id="os" type="text" list="OS">
<datalist id="OS">
    <option value="Mac OS">
    <option value="Windows">
    <option value="Linux">
</datalist>
```

Así se visualizará en Opera:



10. La selección de un color

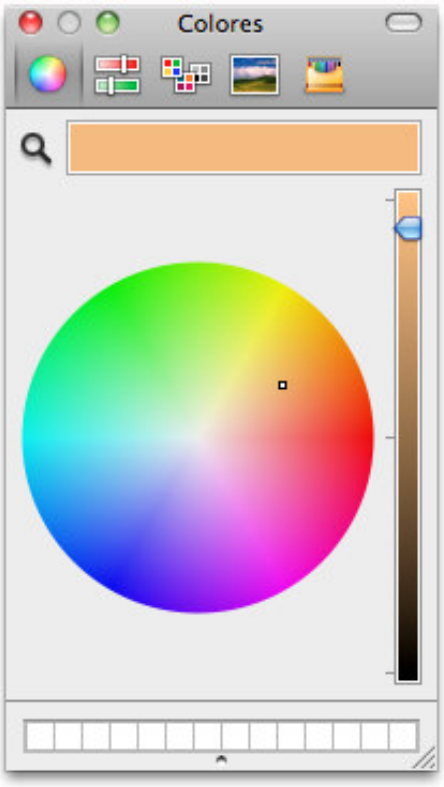
Con este último nuevo elemento para los formularios, los visitantes pueden seleccionar un color en el sistema de selección de colores de su sistema operativo.

Esta sería la sintaxis: `<input type="color">`.

Así se visualizará en Opera:

Si el usuario hace clic en **Otros**, se utilizará el sistema de selección de colores del sistema operativo del internauta.

Ejemplo con Mac OS:



11. Las claves públicas

Si necesita enviar una clave pública con un formulario, utilice el elemento <keygen>.

La validación de los formularios

1. La validación del lado del cliente

Con HTML5 podremos realizar la validación de los datos introducidos en el navegador del lado del cliente. Esto no impide que luego se valide el formulario del lado del servidor, para una mayor seguridad y eficacia. La validación nativa del lado del cliente va a permitir que los diseñadores web no tengan que utilizar JavaScript.

Recuerde que cada navegador podrá elegir con total libertad cómo se va a indicar en pantalla que se ha producido un error.

2. Desactivar la validación

Si así lo desea, puede desactivar sin ningún problema la validación nativa de los formularios en HTML5. Simplemente deberá utilizar el atributo boleano `novalidate` en el elemento `<form>`.

```
<form novalidate="novalidate">...</form>
```

O con la sintaxis abreviada:

```
<form novalidate>...</form>
```

También puede indicarlo en el botón de envío del formulario, añadiendo el atributo `formnovalidate`.

Ejemplo:

```
<input type="submit" id="enviar" value="Enviar" formnovalidate />
```

3. Insertar un campo obligatorio

Para hacer que un campo de introducción de datos sea obligatorio, tendremos que usar el atributo boleano `required`. El navegador no enviará el formulario hasta que el campo obligatorio no haya sido completado.

```
<input type="text" id="nombre" required="required">
```

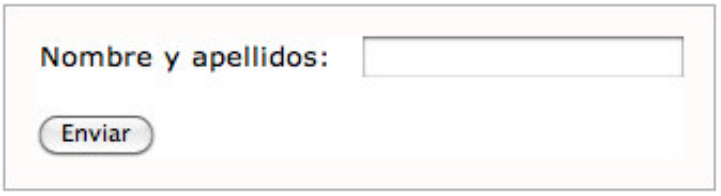
O con la sintaxis abreviada:

```
<input type="text" id="nombre" required>
```

Este es un ejemplo de un campo obligatorio:

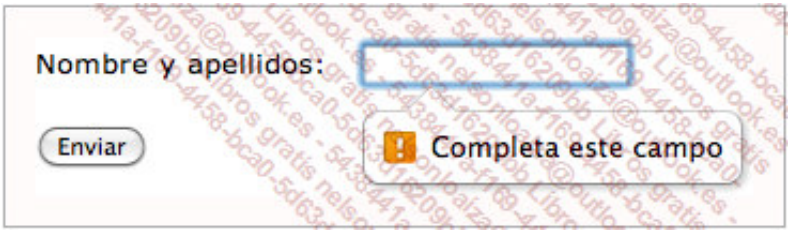
```
<form id="registro" method="#" action="#">
    <p><label for="nombre">Nombre y apellidos: </label>
    <input type="text" id="nombre" required /></p>
    <p><input type="submit" /></p>
</form>
```

La visualización:



Este es el mensaje de aviso que mostrará Opera:

Este es el mensaje de aviso que mostrará Chrome:



4. Los valores autorizados

Usted podrá indicarle al navegador cuáles son los valores autorizados para los campos de inserción de datos. Para ello usaremos el atributo `pattern`, que tendrá como valor una expresión regular. Usted deberá indicar con la expresión regular cuál es el valor autorizado y el navegador mostrará un mensaje de advertencia en caso de error.

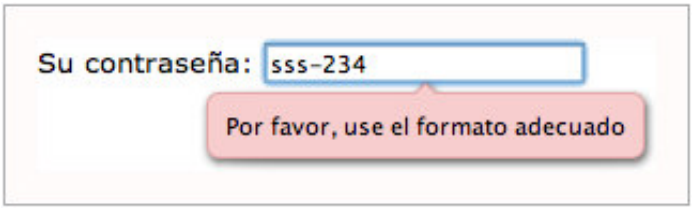
Supongamos que, en un campo determinado, el usuario debe introducir una contraseña con un formato muy estricto: tres cifras, un guión y tres letras en mayúscula solamente.

Esta es la sintaxis que deberá usar:

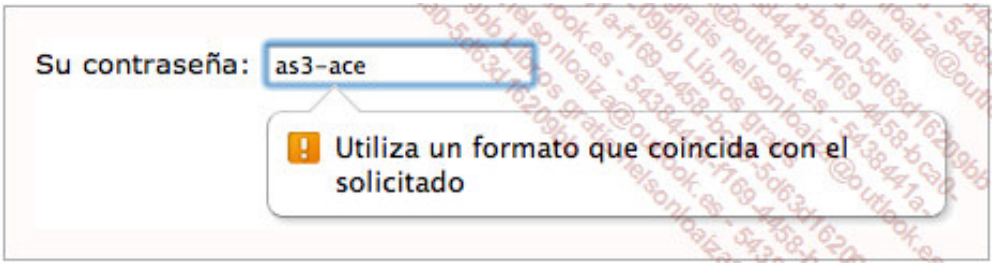
```
<label for="clave">Su contraseña: </label>

<input type="text" pattern="[0-9]{3}-[A-Z]{3}" id="clave">
```

- `[0-9]`: indica que se admiten todas las cifras de 0 a 9.
- `{3}`: indica que es obligatorio introducir tres cifras.
- `-`: indica que se debe insertar el carácter -.
- `[A-Z]`: indica que se admiten todas las letras en mayúsculas de A a Z.
- `{3}`: indica que es obligatorio introducir tres letras.

Este es el mensaje de advertencia que mostrará Opera cuando el valor introducido no sea conforme:



Este es el mensaje de advertencia que mostrará Chrome:



Las expresiones regulares

En el ejemplo anterior vimos cómo usar una expresión regular en el atributo `pattern`. Estudiemos más detenidamente esas expresiones regulares.

Estas son algunas URL sobre el tema:

- http://es.wikipedia.org/wiki/Expresi%C3%B3n_regular
- http://www.desarrolloweb.com/manuales/expresiones-regulares.html
- http://html5pattern.com/
- http://www.w3schools.com/jsref/jsref_obj_regexp.asp
- http://blog.stevenlevithan.com/

1. Las letras autorizadas

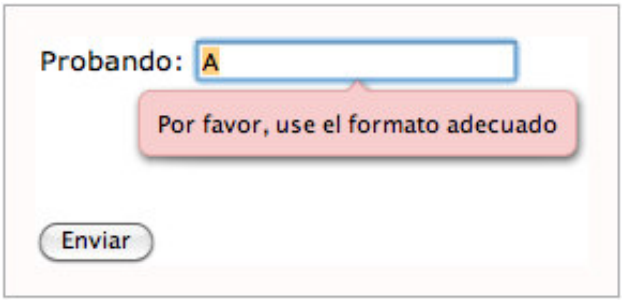
Esta es la sintaxis para indicar que el usuario solamente podrá insertar una letra `a` o `g` o `r` y que esta deberá estar en minúsculas: `[agr]`. Cualquier otro valor no será válido.

Esta sería la sintaxis:

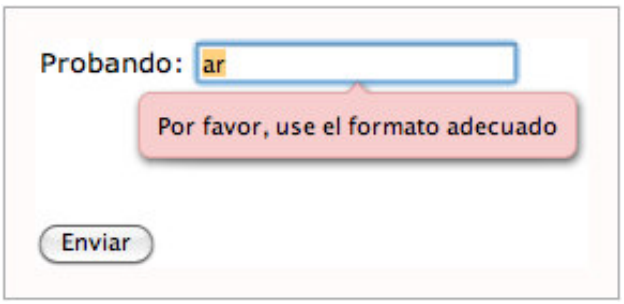
```
<input type="text" id="test" pattern="[agr]" />
```

Ejemplos de valores no válidos:

`A` está en mayúscula:



Hay dos caracteres:

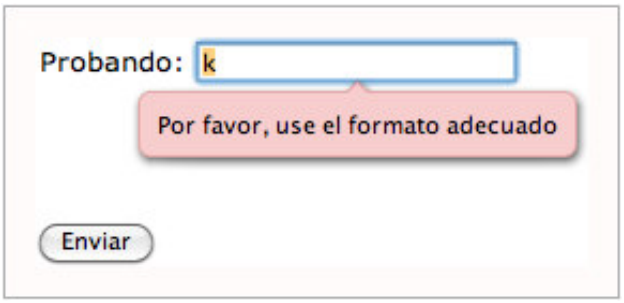


2. Un intervalo de letras autorizadas

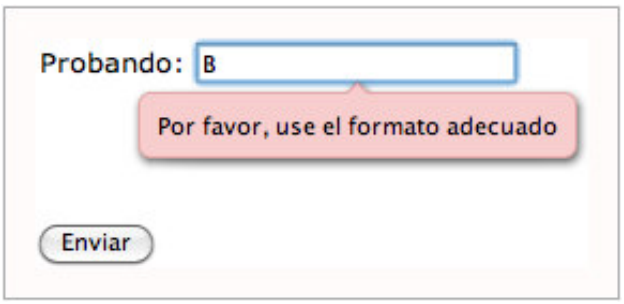
Con esta sintaxis podrá indicar un intervalo de letras autorizadas: `[G-M]`. El usuario podrá indicar una letra mayúscula comprendida entre `G` y `M`, ambas incluidas.

Ejemplos de valores no válidos:

`k` está en minúscula:



`B` no pertenece al intervalo:

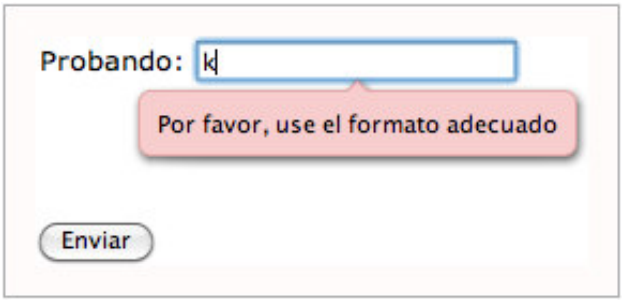


3. Otras letras a excepción de

Esta es la sintaxis para indicar que todas las letras están autorizadas, a excepción de las indicadas: `[^abc]`. El usuario podrá usar todos los caracteres salvo `a`, `b` y `c`.

Esta es la sintaxis para indicar que todas las letras están autorizadas, a excepción de las indicadas en el intervalo: `[^d-m]`. El usuario podrá usar todas las letras a excepción de las comprendidas entre `d` y `m`, ambas incluidas.

`k` está dentro del intervalo no autorizado:



4. Las mayúsculas y minúsculas

Usted puede exigir que las letras autorizadas estén en mayúsculas o en minúsculas.

Esta es la sintaxis para indicar que se puede introducir un único carácter en mayúscula: `[A-Z]`.

Esta es la sintaxis para indicar que se puede introducir un único carácter en minúscula: `[a-z]`.

Esta es la sintaxis para indicar que se puede introducir un único carácter en mayúscula o en minúscula: `[A-z]`.

5. La condición O

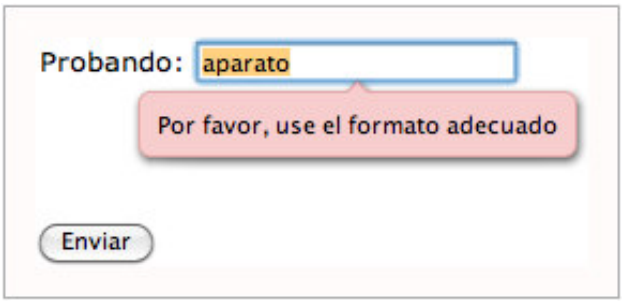
Es posible expresar varias posibilidades con el carácter `|`, que representa la función lógica `O`.

Esta es la sintaxis para indicar que se puede introducir un carácter comprendido entre `a` y `e` o `w` y `z`: `[a-e|w-z]`.

6. Las palabras autorizadas

También podrá indicar qué palabras están autorizadas. Esta es la sintaxis para indicar que se pueden introducir las palabras `cosa` o `chisme` o `cacharro`: `(cosa|chisme|cacharro)`.

`aparato` no está autorizado:

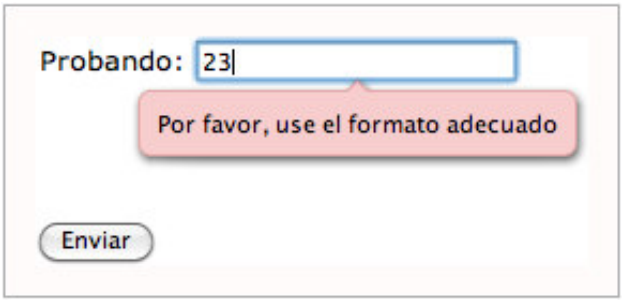


7. Autorizar cifras

Encontramos exactamente el mismo principio para las cifras.

Esta es la sintaxis para indicar que se puede introducir una cifra comprendida entre `0` y `9`: `[0-9]`.

23 no está autorizado, se trata de un número, y no de una simple cifra:



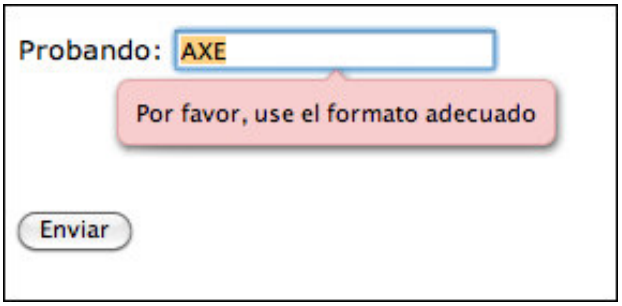
8. Un número limitado de caracteres

Puede indicar cuántas letras o cifras se admiten para que el valor introducido sea válido.

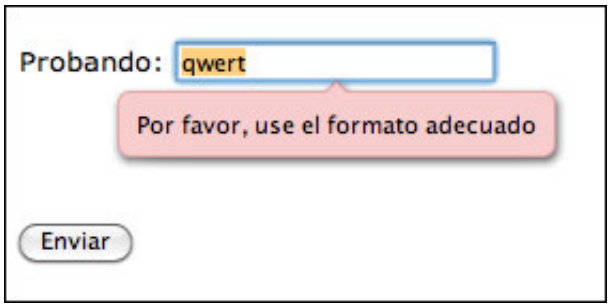
Esta es la sintaxis para indicar que se pueden introducir `5` letras en mayúsculas: `[A-Z]{5}`.

Ejemplos de valores no válidos:

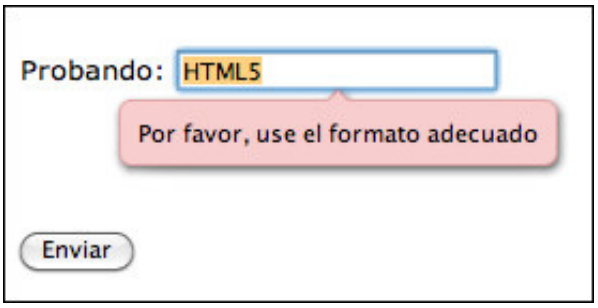
Solamente hay tres letras:



Efectivamente hay cinco letras, pero están en minúsculas:



Hay cuatro mayúsculas y una cifra:

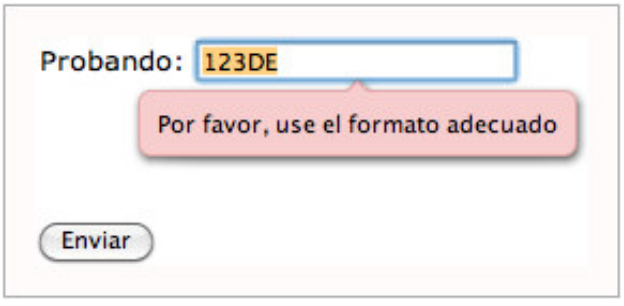


9. Los valores múltiples

Esta es la sintaxis para indicar que se puede introducir un valor que incluya dos letras en mayúsculas seguidas de tres cifras: `[A-Z]{2}[0-9]{3}`.

Ejemplo de inserción no válida:

Efectivamente se ha introducido el número de caracteres correcto, pero se ha comenzado por las cifras y no por las letras:



10. Los parámetros especiales

Podemos usar algunos caracteres especiales para definir determinadas expresiones regulares.

- `\d` hace referencia a las cifras. `\d{3}` admite la introducción de `3` cifras.
- `\w` hace referencia a los caracteres. `\w{3}` admite la introducción de `3` caracteres.
- `+` indica que debe haber uno o más caracteres de los especificados anteriormente. De este modo el valor no estará limitado. `[a-z]+` indica que el usuario podrá insertar tantas letras minúsculas como desee.
- `*` indica que pueden haber varios o ningún carácter de los especificados anteriormente. De este modo podemos obtener valores opcionales. `[A-Z]{2}[0-9]*` indica que el valor comenzará con dos mayúsculas que podrán ir seguidas de cero o n cifras.
- `^` indica que el valor deberá comenzar obligatoriamente por los elementos especificados justo a continuación. `^[0-9]{2}` indica que el valor deberá comenzar con dos cifras para que sea válido.
- `$` indica que el valor deberá terminar obligatoriamente por los elementos específicos que la preceden. `[a-z]$` indica que el valor deberá terminar con una letra en minúscula para que sea válido.

Esta es la sintaxis que permite autorizar un valor entre 20 y 29: `^(2[0-9])$`. El valor deberá comenzar por 2 (`^(2`) y terminar con una cifra comprendida entre 0 y 9 (`[0-9])$`).

Las ayudas para los usuarios de un formulario

1. La ayuda para completar

El atributo `placeholder` permite mostrar un valor de ejemplo en un campo de texto para ayudar al usuario. Este verá así un ejemplo del tipo de valor que tiene que insertar. En cuanto el usuario haga clic en el campo, el valor de ejemplo desaparecerá.

Esta es la sintaxis que podríamos usar, si volvemos a tomar el ejemplo anterior:

```
<input type="text" pattern="[0-9]{3}-[A-Z]{3}"
placeholder="111-AAA" id="clave">
```

Se visualizará así cuando el usuario no haya hecho clic en el campo:

Se visualizará así cuando el usuario haya hecho clic en el campo:

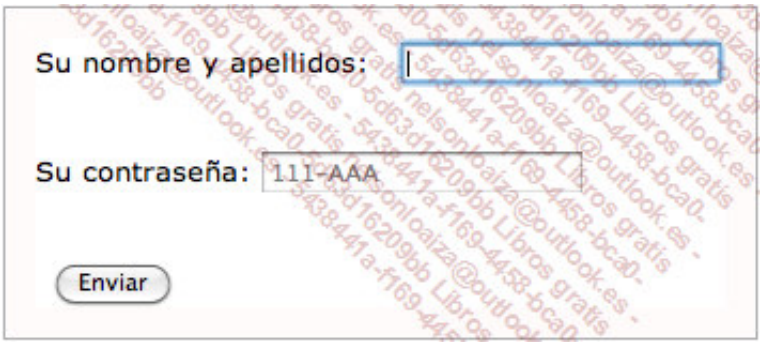
2. Activar un campo

Usted puede hacer que el punto de inserción parpadee en un campo específico utilizando el atributo boleano `autofocus`.

Veamos un ejemplo en el que se ha usado este atributo con una sintaxis abreviada:

```
<label for="nombre">Su nombre y apellidos: </label>
<input type="text" id="nombre" autofocus />
```

Cuando se abra la página, verá que el punto de inserción parpadea en el interior del campo seleccionado:



Deberá usar un único `autofocus` en cada página.

Utilice esta funcionalidad con "moderación": algunos internautas la "odian", porque sienten que se les está obligando a insertar un valor en dicho campo. Además, algunos usuarios experimentados utilizan la barra espaciadora para recorrer las páginas web y, con el autofocus, no será posible usar

esa funcionalidad. Aunque, "en principio", los navegadores deberían disponer de una opción para desactivar el autofocus.

3. El completado automático

En los formularios, los navegadores usan de manera predeterminada el atributo `autocomplete`con el valor `on`, lo que permite acceder a la lista de los últimos valores introducidos en un campo. De este modo es posible seleccionar rápidamente un valor que ya hayamos insertado con anterioridad.

```
<form id="registro" method="post" action="registro.php"
autocomplete="on">
```

Aunque usted haya activado la funcionalidad de autocompletado para la totalidad de los campos de un formulario, no tiene por qué usarla, si no lo desea, en un campo determinado:

```
<input type="url" id="url" autocomplete="off" />
```

4. Redimensionar un campo

La nueva propiedad CSS3 `resize` permite autorizar a los usuarios para que puedan cambiar la dimensión de un campo de texto. Esta propiedad admite cinco valores:

- `none`: no se autoriza la redimensión.
- `both`: se autoriza la redimensión vertical y horizontal.
- `horizontal`: solamente se autoriza el redimensión horizontal.
- `vertical`: solamente se autoriza el redimensión vertical.
- `inherit`: hereda la propiedad del elemento padre.

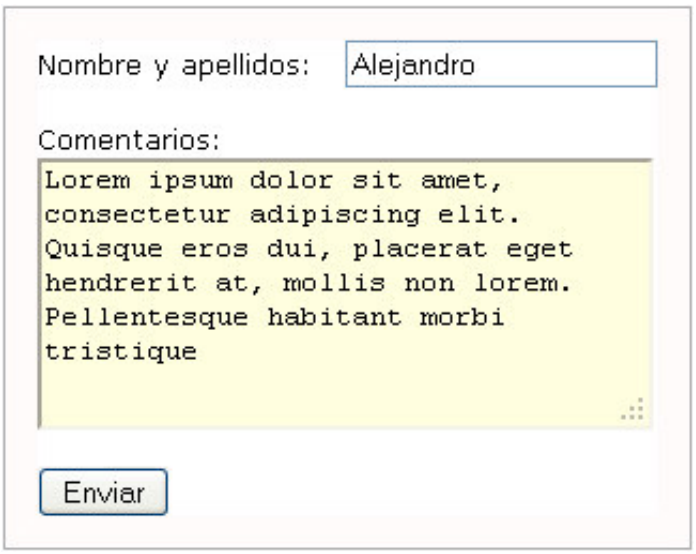
Veamos el formulario:

```
<form id="test" method="#" action="#">
    <p>
        <label for="nombre">Nombre y apellidos: </label>
        <input type="text" id="nombre" />
    </p>
    <p>
        <label for="comentario">Comentarios: </label>
        <br />
        <textarea name="comentario" id="comentario"></textarea>
    </p>
    <p>
        <input type="submit" name="enviar" id="enviar" value="Enviar" />
    </p>
</form>
```

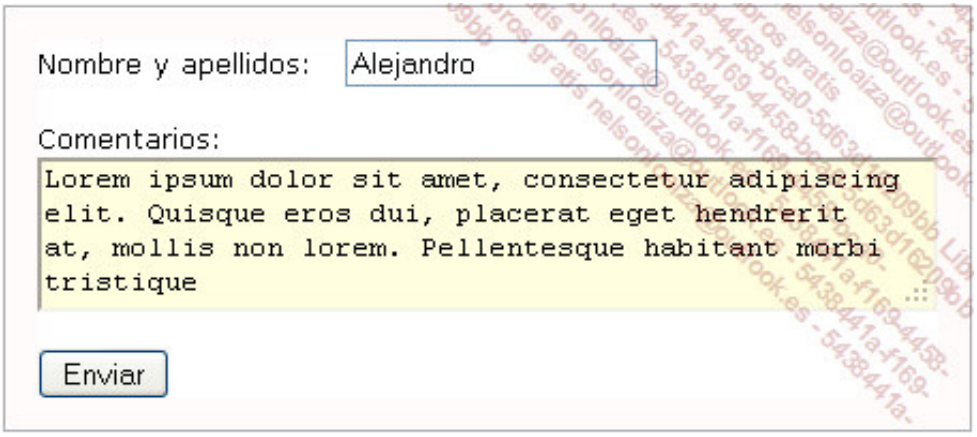
Veamos el estilo CSS para el elemento `<textarea>`:

```
#comentario{
    width: 200px;
    height: 90px;
    border: 1px solid #333;
    background-color: lightyellow;
    resize: both;
}
```

Así se visualizará con Firefox:



Se puede ajustar el tamaño tanto vertical como horizontalmente:



Las pseudo-clases para los formularios

1. Las funcionalidades de las pseudo-clases

CSS3 introduce novedades muy interesantes con las pseudo-clases específicas para los elementos de los formularios. Vamos a poder aplicar estilos diferentes a los elementos activos de un formulario, los elementos deshabilitados y los elementos seleccionados. De este modo el usuario verá lo que ha hecho y qué elementos están habilitados o deshabilitados.

Veamos esas nuevas pseudo-clases: `:enabled`, `:disabled` y `:checked`.

2. El formulario y los estilos CSS

A continuación tenemos un formulario con dos campos de inserción de datos: el primer campo, que permite insertar el nombre, está habilitado (`enabled`); el segundo campo, que sirve para insertar la edad, está deshabilitado (`disabled`).

Luego tenemos dos botones de opción para el sexo (hombre o mujer).

Y, por último, una casilla de selección.

Vamos ahora a diferenciar visualmente los elementos del formulario en función del estado: habilitado, deshabilitado o seleccionado.

Este es el código HTML del formulario:

```
<form id="form1" method="#" action="#">
<p>
     <label for="nombre">Su nombre y apellidos: </label>
     <input type="text" id="nombre" />
</p>
<p>
     <label for="edad">Su edad: </label>
     <input type="text" id="edad" disabled="disabled"/>
</p>
<p>
     <input type="radio" name="sexo" id="hombre" value="hombre" />
     <label for="hombre">hombre</label><br/>
     <input type="radio" name="sexo" id="mujer" value="mujer" />
     <label for="mujer">mujer</label>
</p>
<p>
     <input type="checkbox" id="acepto" />
     <label for="acepto">Acepto las condiciones de uso.</label>
</p>
</form>
```

Y estos son los estilos CSS:

```
#acepto:checked+label {
    background-color: gold;
    font-weight: bold;
}
#hombre:checked+label, #mujer:checked+label {
    font-style: italic;
}
:enabled {
    background-color: lightgreen;
}
:disabled {
```

```
    background-color: lightcoral;
}
```

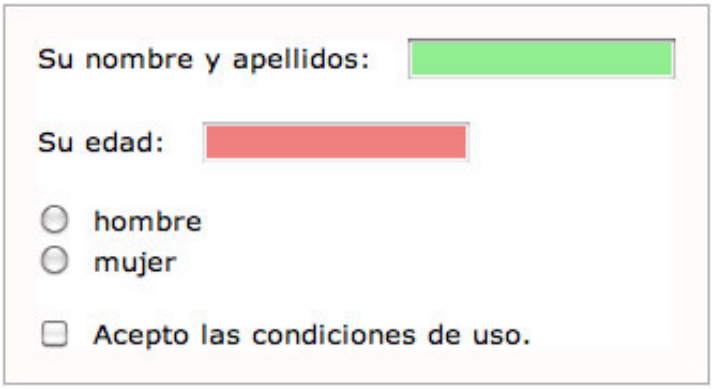
El primer selector se dirige a la etiqueta (`label`) de la casilla de selección (`#acepto`) y se aplicará cuando esta aparezca seleccionada (`checked`). Claro está, nos gustaría resaltar la etiqueta de la casilla, y no la casilla, por lo que hemos usado un selector adyacente.

El segundo selector se dirige al botón de opción seleccionado.

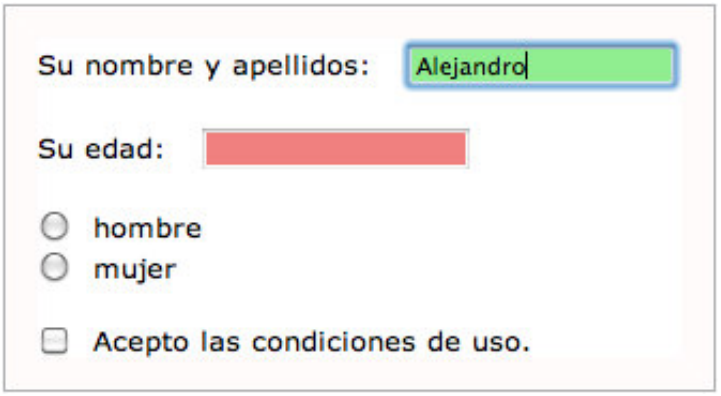
Los otros dos estilos añaden, simplemente, un fondo de color en función del estado del campo.

3. Resultado

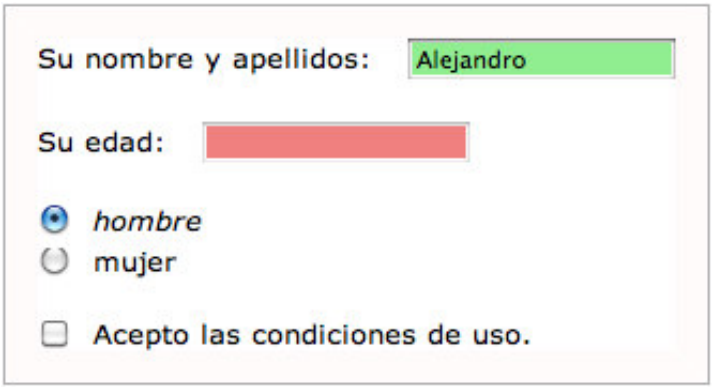
Así se visualizará el formulario cuando el usuario no haya realizado ninguna acción:



El usuario podrá introducir un valor en el primer campo activo (con fondo verde), pero no en el segundo, que se encuentra deshabilitado (fondo rojo):

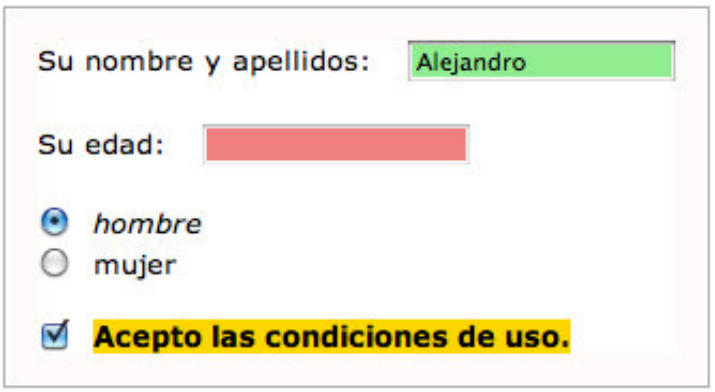


El usuario ha seleccionado un botón de opción, así que su etiqueta aparece en cursiva:



El usuario ha activado la casilla de selección, así que la etiqueta aparece en negrita con un fondo

dorado:



4. Otra pseudo-clase

Existe otro estado para los botones de opción y las casillas de selección, se trata del estado intermedio: `:indeterminate` . Este estado indica que el elemento no está ni seleccionado, ni no seleccionado.

5. Los campos obligatorios y facultativos

Usted puede cambiar el diseño de los campos obligatorios y facultativos. Utilice para ello las pseudo-clases `:required` y `:optional`.

Los campos obligatorios, que requieren que se haya insertado un valor, deben presentar el atributo boleano `required`. Los que no sean obligatorios, serán por defecto facultativos, sin que haga falta precisarlo.

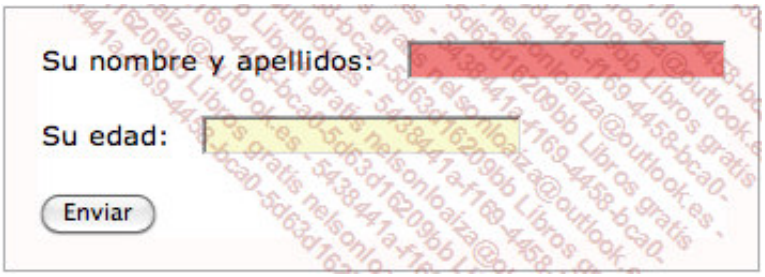
Veamos un ejemplo de formulario:

```
<form id="form1" method="#" action="#">
    <p>
        <label for="nombre">Su nombre y apellidos: </label>
        <input type="text" name="nombre" id="nombre" required />
    </p>
    <p>
        <label for="edad">Su edad: </label>
        <input type="text" name="edad" id="edad" />
    </p>
    <p>
        <input type="submit" id="enviar" value="Enviar" />
    </p>
</form>
```

Y estos son los estilos CSS:

```
input:required {
    background-color: lightcoral;
}
input:optional {
    background-color: lightgoldenrodyellow;
}
input#enviar {
    background: none;
}
```

Y el resultado visual obtenido:



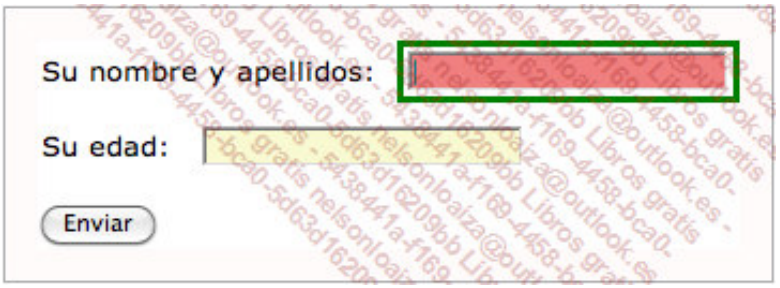
6. El formato de :focus

La pseudo-clase `:focus` permite resaltar el campo que se esté usando en ese momento con un borde, `outline`. Los estilos CSS3 introducen una nueva propiedad, `outline-offset`, que corresponde a la distancia entre el límite de la caja y el límite del borde.

Veamos un ejemplo muy sencillo:

```
input:focus {
   outline: solid 3px green;
   outline-offset: 2px;
}
```

Y el resultado visual:



La validación de los datos introducidos

1. Funcionalidad

Al principio del capítulo, en la sección La validación de los formularios, vimos cómo podíamos hacer que un campo fuese obligatorio con el atributo `required` y cuál era la reacción por defecto de los navegadores. Ahora vamos a aprender a modificar el diseño de la validación de los formularios con los estilos CSS3.

2. El formulario

Vamos a crear un formulario con tres campos y un botón de envío.

Este es el código HTML del formulario:

```
<form id="registro" method="#" action="#">
   <p>
       <label for="nombre">Su nombre: </label>
       <input type="text" id="nombre" required />
   </p>
   <p>
       <label for="email">Su e-mail: </label>
       <input type="email" id="email" required />
   </p>
   <p>
       <label for="edad">Su edad: </label>
       <input type="text" id="edad" pattern="\d{2}" />
   </p>
   <p>
       <input type="submit" id="enviar" value="Enviar" />
   </p>
</form>
```

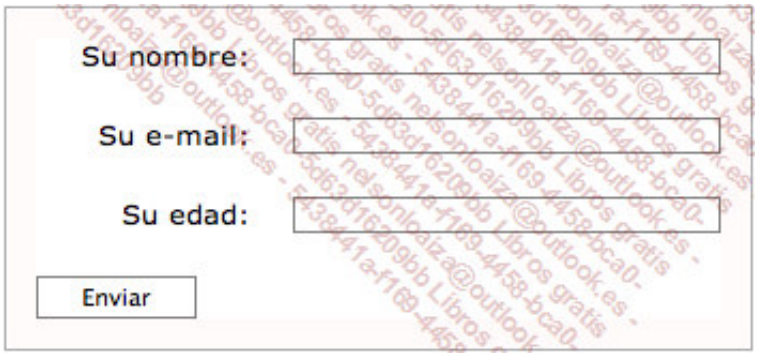
Los dos primeros campos son obligatorios (`required`) y el último incluye una validación con una expresión regular. Se ha insertado un único botón de envío.

3. El diseño del formulario

Estos son los estilos que se han usado para aplicar un diseño sencillo a los elementos del formulario:

```
input {
    border: solid 1px gray;
    width: 200px;
}
label, input{
    display: inline-block;
}
label {
    width: 100px;
    text-align: right;
    margin-right: 10px;
}
input[type=submit]{
    width: 75px;
    background: none;
}
```

Este es el resultado visual en Opera:



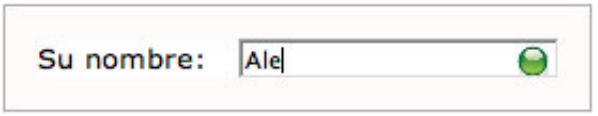
4. Los campos no válidos

A continuación tenemos un campo de tipo `email` y un campo con un `pattern`. La validación de los campos se realiza de forma continua, constantemente. Vamos a ver cómo funciona la validación cuando el campo está activo, con el elemento `:focus` y las pseudo-clases `:valid` y `:invalid`. Se ha establecido que aparezca un punto rojo a la derecha del campo cuando el valor no sea válido y un punto verde cuando dicho valor sea válido.

```
input:focus:invalid {
    background: url(rojo.png) no-repeat 98% center;
}
input:focus:valid {
    background: url(verde.png) no-repeat 98% center;
}
```

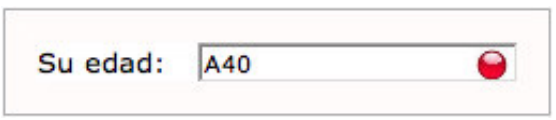
Cuando el usuario hace clic en el campo **Su nombre**, el valor no es válido aún, ya que por el momento no se ha insertado ningún valor. El campo está vacío y, claro, se trata de un campo obligatorio. Por ese motivo aparece el punto rojo.

En cuanto el usuario comienza a insertar caracteres, el valor es validado y aparece el punto verde.

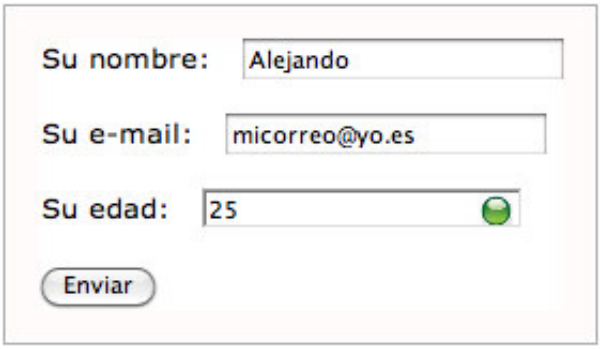


La validación del e-mail sigue el mismo principio. Mientras el símbolo arroba no se encuentre entre los caracteres insertados, no se validará el valor del campo.

Teniendo en cuenta el *pattern* del campo de la edad, si se inserta un valor que no sean dos cifras, no se validará.



En la siguiente captura, todos los valores insertados son válidos:

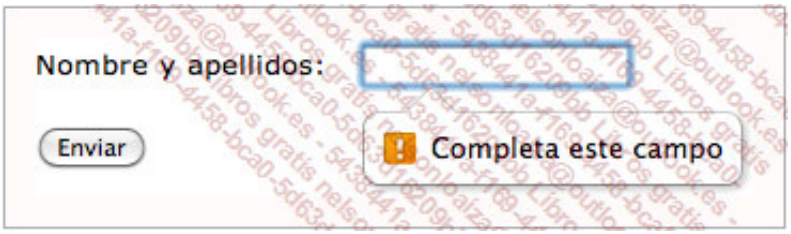


Por último, para evitar que el punto verde aparezca junto al botón de envío cuando se haga clic, utilice la pseudo-clase `:not` para excluirlo mediante su código de identificación.

```
input:focus:valid:not(#enviar) {
    background: url(verde.png) no-repeat 98% center;
}
```

5. Los mensajes de error

Como usted sabe, los navegadores utilizan sus propios elementos de interfaz para mostrar los mensajes de error que aparecen cuando el valor insertado no es válido. Un ejemplo con Chrome:



Vamos ahora a personalizar ese mensaje de error.

En nuestro formulario vamos a añadir el elemento `<span>` inmediatamente después del elemento`<input>` del primer campo **Su nombre y apellidos**. El contenido de `<span>` es el texto que se mostrará cuando el valor introducido no sea válido.

```
<p>
    <label for="nombre">Su nombre y apellidos: </label>
    <input type="text" id="nombre" required />
    <span class="noValido">Este campo no puede dejarse vacío</span>
</p>
```

Habrá notado que `<span>` usa la clase `noValido`.

Veamos los estilos CSS que le hemos aplicado a la clase `noValido` :

```
span.noValido {
    display: none;
    position: relative;
    left: 250px;
    top: 5px;
    width: 230px;
    border: solid 1px red;
    background-color: lightgoldenrodyellow;
    padding: 3px;
```

```
    text-align: center;
}
```

La propiedad importante es la visualización, `display`, que se encuentra en `none`, para no mostrar el mensaje de error. Todo lo demás es la aplicación de las propiedades habituales.

Ahora tenemos que hacer que aparezca ese mensaje si se produce un error, cuando el campo esté activo.

Vamos a usar el selector:

```
input:focus:invalid + span.noValido {
    display: block;
}
```

- `input`: para un elemento de formulario de tipo campo de inserción de datos.
- `:focus`: esta pseudo-clase indica que el campo está activo.
- `:invalid`: esta pseudo-clase indica que el campo no es válido.
- `+` es el selector de adyacente directo que seleccionará al hermano inmediato de un elemento. En nuestro ejemplo, seleccionará el elemento inmediatamente después del campo `input`, es decir, el elemento `span` (como puede comprobar en el código facilitado anteriormente).
- `span.noValido`: se aplica a los elementos `<span>` que utilizan la clase `noValido`.

Hemos usado la propiedad `display` en `block` para mostrar el mensaje con el diseño que acabamos de establecer.

Veamos cómo funciona el mensaje de advertencia:

El mensaje aparecerá cuando el usuario no haya introducido ningún dato en el campo **Su nombre y apellidos**.

En cuanto introduzca un carácter, el mensaje se ocultará.

Seguiremos el mismo principio para el campo de la dirección de e-mail:

```
<p>
   <label for="email">Su e-mail: </label>
   <input type="email" id="email" required />
   <span class="noValido">La dirección de e-mail no es correcta</span>
</p>
```

El mensaje de error para la dirección de e-mail:

6. Los campos válidos

Ahora queremos indicarle a los usuarios que el valor que han insertado es válido. Vamos a tomar como ejemplo el campo e-mail.

Añadamos un elemento para insertar el mensaje de validación.

```
<p>
   <label for="email">Su e-mail: </label>
   <input type="email" id="email" required />
   <span class="noValido">La dirección de e-mail no es correcta</span>
   <span class="valido">La dirección de e-mail es correcta</span>
</p>
```

Añadamos los estilos CSS para el formato del mensaje de validación:

```
span.valido {
    display: none;
    position: relative;
    left: 250px;
    top: 5px;
    width: 230px;
    border: solid 1px green;
    background-color: lightgreen;
    padding: 3px;
    text-align: center;
}
```

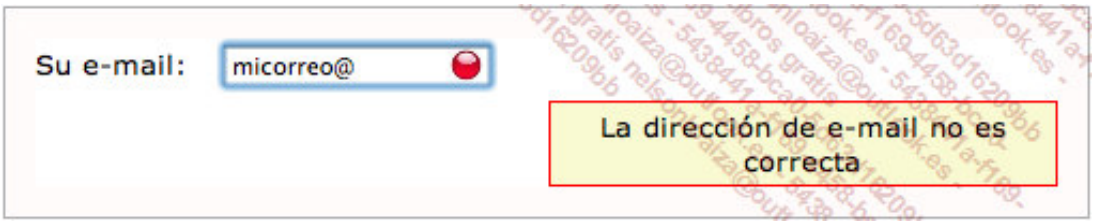
Ahora vamos a hacer que aparezca el mensaje de validación. En la sintaxis anterior habíamos utilizado el selector de adyacente directo: +, pero ahora tenemos dos detrás de <input>. Así que esta vez vamos a usar el selector adyacente general ~.

El estilo CSS modificado sería:

```
input:focus:invalid ~ span.noValido, input:focus:valid ~
span.valido {
    display: block;
}
```

Veamos cómo funciona el campo de inserción de la dirección de e-mail:

Este campo, de momento, no es válido, así que se muestra el mensaje de error:



Ahora la dirección de e-mail es válida, así que se muestra el mensaje de validación:

7. Los valores fuera de los límites

Disponemos de otras dos pseudo-clases para validar los números y las fechas. La pseudo-clase `:in-range` se aplica a los valores que se encuentran dentro de los límites definidos y la pseudo-clase `:out-of-range` se aplica a los valores que excedan de esos límites.

Veamos una aplicación sencilla con un formulario que contiene un campo de tipo `number`, con un límite inferior de `18` y un límite superior de `30`:

```
<form id="test" method="#" action="#">
<p>
   <label for="edad">Indique su edad: </label>
    <input type="number" min="18" max="30" step="2" value="25" id="edad" />
</p>
<p>
   <input type="submit" id="enviar" value="Enviar" />
</p>
</form>
```

Estos son los estilos CSS:

```
<style>
    input#edad:out-of-range {
        background-color: lightpink;
    }
    input#edad:in-range {
        background-color: lightgreen;
    }
</style>
```

Y este sería el resultado visual:

La edad indicada se encuentra dentro de los límites, así que se muestra un fondo verde:

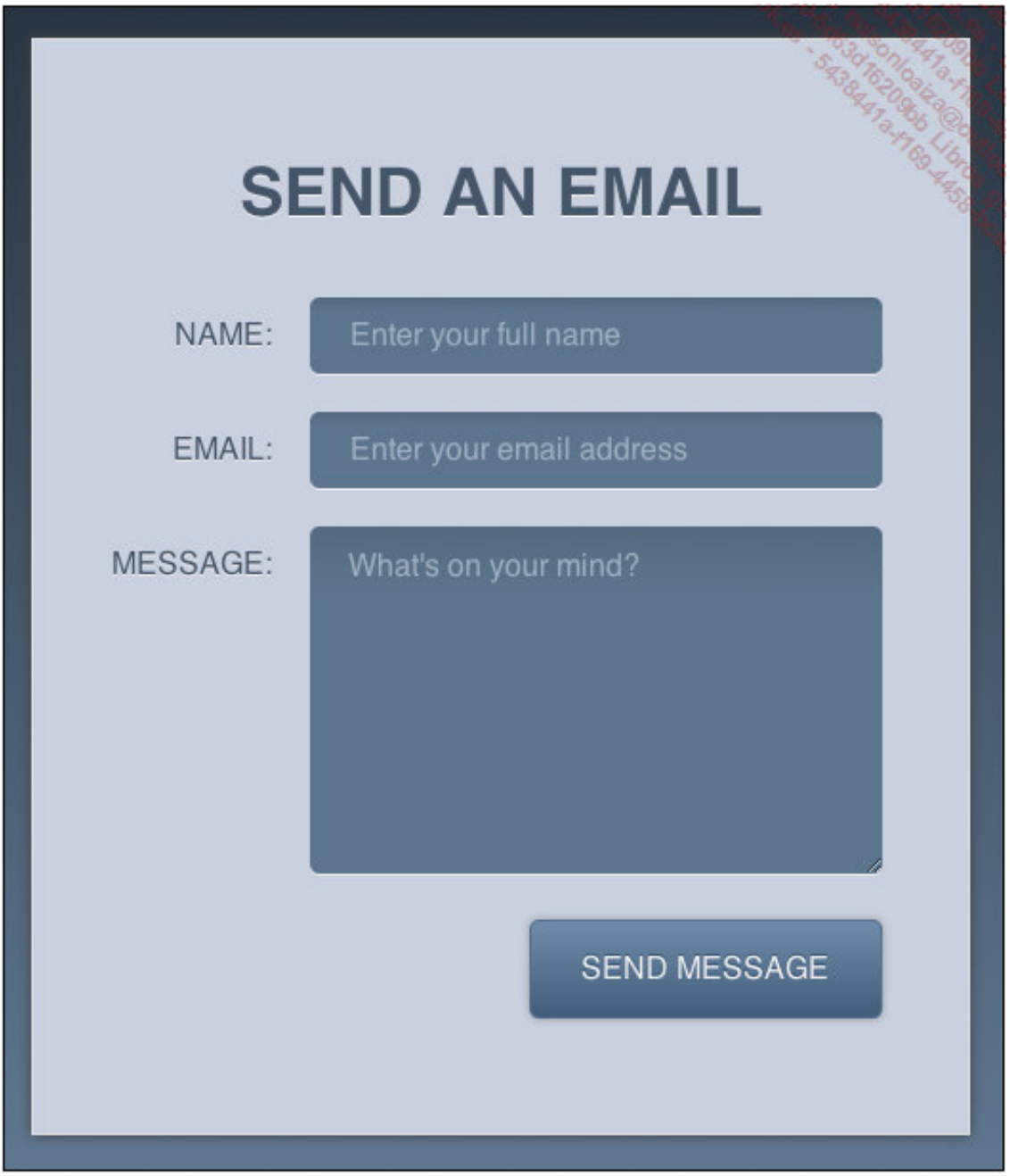
La edad indicada se encuentra fuera de los límites, así que se muestra un fondo rosa:

Un primer ejemplo de formulario sencillo

1. Objetivo

Vamos a examinar cómo ha sido creado el primer ejemplo de formulario simple en HTML5/CSS3 que se encuentra disponible en la URL: http://line25.com/tutorials/create-a-stylish-contact-form-with-html5-css3

Este es el resultado final:



2. La estructura del formulario

El formulario se encuentra dentro de una caja `<div id="content">` para la aplicación de los estilos. Los cuatro objetos del formulario se han insertado en elementos `<fieldset>`.

Los dos primeros campos son elementos `<input>` de tipo `text` y `email`. Todos los elementos presentan el atributo `placeholder` para mostrar un valor inicial de ejemplo en la zona de

inserción de datos.

El campo de inserción multilingüe es un elemento `<textarea>` con el atributo `placeholder`.

El botón de envío del formulario es un botón clásico.

Este es el código del formulario:

```
<div id="contact">
   <h1>Send an email</h1>
   <form action="#" method="post">
       <fieldset>
          <label for="name">Name:</label>
          <input type="text" id="name" placeholder="Enter
your full name" />
          <label for="email">Email:</label>
          <input type="email" id="email" placeholder="Enter
your email address" />
          <label for="message">Message:</label>
          <textarea id="message" placeholder="What's on your
mind?"></textarea>
          <input type="submit" value="Send message" />
       </fieldset>
   </form>
</div>
```

3. Los estilos generales

En un primer momento, el diseñador web especifica las características de los elementos presentes en la página, atribuyendo el valor `0` a sus propiedades.

```
body, div, h1, form, fieldset, input, textarea {
    margin: 0;
    padding: 0;
    border: 0;
    outline: none;
}
html {
    height: 100%;
}
```

4. El fondo de la página

El fondo de la página es un degradado de color aplicado a elemento `body`, con las adaptaciones para los diferentes navegadores:

```
body {
    background: #728eaa;
    background: -moz-linear-gradient(top, #25303C 0%, #728EAA
100%); /* firefox */
    background: -webkit-gradient(linear, left top, left bottom,
color-stop(0%,#25303C), color-stop(100%,#728EAA)); /* webkit */
    font-family: sans-serif;
}
```

5. El formulario

El formulario ha sido insertado en una caja `<div id="content">`. Esta caja `<div>` dispone de

un ancho, de márgenes, de espaciado, de un color de fondo y de un borde. Además se le ha aplicado una sombra de caja (`box-shadow`) para los distintos navegadores.

```
#contact {
    width: 430px;
    margin: 60px auto;
    padding: 60px 30px;
    background: #c9d0de;
    border: 1px solid #e1e1e1;
    -moz-box-shadow: 0px 0px 8px #444;
    -webkit-box-shadow: 0px 0px 8px #444;
}
```

El título es un simple `h1` con una sombra:

```
h1 {
    font-size: 35px;
    color: #445668;
    text-transform: uppercase;
    text-align: center;
    margin: 0 0 35px 0;
    text-shadow: 0px 1px 0px #f2f2f2;
}
```

6. Las etiquetas

Las etiquetas de los campos `label` presentan, entre otros estilos, una sombra:

```
label {
    float: left;
    clear: left;
    margin: 11px 20px 0 0;
    width: 95px;
    text-align: right;
    font-size: 16px;
    color: #445668;
    text-transform: uppercase;
    text-shadow: 0px 1px 0px #f2f2f2;
}
```

7. Los campos de inserción de datos

Los campos `input` presentan, en especial, degradados lineales, bordes redondeados, sombras aplicadas a los distintos elementos y al texto.

```
input {
    width: 260px;
    height: 35px;
    padding: 5px 20px 0px 20px;
    margin: 0 0 20px 0;
    background: #5E768D;
    background: -moz-linear-gradient(top, #546A7F 0%, #5E768D
20%); /* firefox */
    background: -webkit-gradient(linear, left top, left bottom,
color-stop(0%,#546A7F), color-stop(20%,#5E768D)); /* webkit */
    border-radius: 5px;
    -moz-border-radius: 5px;
    -webkit-border-radius: 5px;
    -moz-box-shadow: 0px 1px 0px #f2f2f2;
    -webkit-box-shadow: 0px 1px 0px #f2f2f2;
    font-family: sans-serif;
    font-size: 16px;
    color: #f2f2f2;
    text-transform: uppercase;
    text-shadow: 0px -1px 0px #334f71;
}
```

Los valores de ejemplo (`placeholder`), también presentan estilos de formato, en especial, texto sombreado:

```
input:-webkit-input-placeholder  {
    color: #a1b2c3;
    text-shadow: 0px -1px 0px #38506b;
}

input:-moz-placeholder {
    color: #a1b2c3;
    text-shadow: 0px -1px 0px #38506b;
}
```

8. El campo multilínea

El campo de texto multilínea `textarea` presenta las propiedades CSS3 de degradado, bordes redondeados y una sombra bajo el texto.

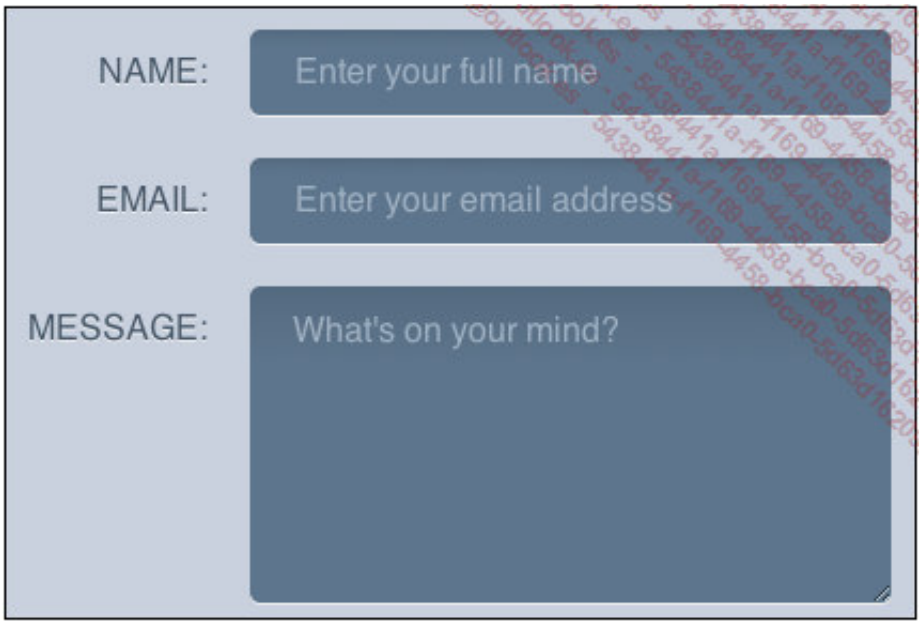
```
textarea {
    width: 260px;
    height: 170px;
    padding: 12px 20px 0px 20px;
    margin: 0 0 20px 0;
    background: #5E768D;
    background: -moz-linear-gradient(top, #546A7F 0%, #5E768D
20%); /* firefox */
    background: -webkit-gradient(linear, left top, left bottom,
color-stop(0%,#546A7F), color-stop(20%,#5E768D)); /* webkit */
    border-radius: 5px;
    -moz-border-radius: 5px;
    -webkit-border-radius: 5px;
    -moz-box-shadow: 0px 1px 0px #f2f2f2;
    -webkit-box-shadow: 0px 1px 0px #f2f2f2;
    font-family: sans-serif;
```

```
    font-size: 16px;
    color: #f2f2f2;
    text-transform: uppercase;
    text-shadow: 0px -1px 0px #334f71;
}
```

Los ejemplos de valores posibles (`placeholder`), también presentan estilos de formato y, en especial, texto sombreado.

```
textarea::-webkit-input-placeholder  {
    color: #a1b2c3;
    text-shadow: 0px -1px 0px #38506b;
}
textarea:-moz-placeholder {
    color: #a1b2c3;
    text-shadow: 0px -1px 0px #38506b;
}
```

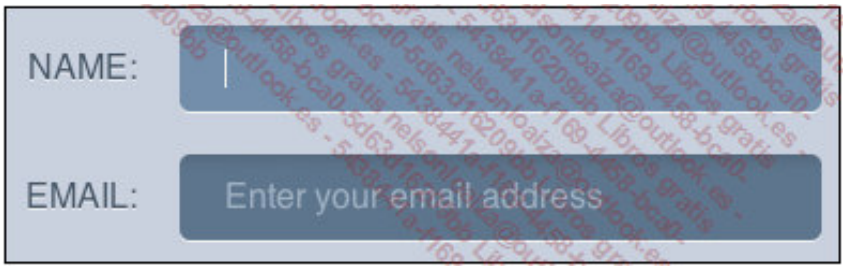
Este es el resultado visual de los estilos anteriores:



9. Los campos activos

Los campos de inserción de datos (`input` y `textarea`) tendrán un formato diferente cuando se estén usando, mediante la pseudo-clase `:focus`: se les aplicará un degradado.

```
input:focus, textarea:focus {
    background: #728eaa;
    background: -moz-linear-gradient(top, #668099 0%, #728eaa
20%); /* firefox */
    background: -webkit-gradient(linear, left top, left bottom,
color-stop(0%,#668099), color-stop(20%,#728eaa)); /* webkit */
}
```



10. El botón de envío

El botón de envío (`input[type=submit]`) presenta como estilos CSS3 una sombra y un degradado.

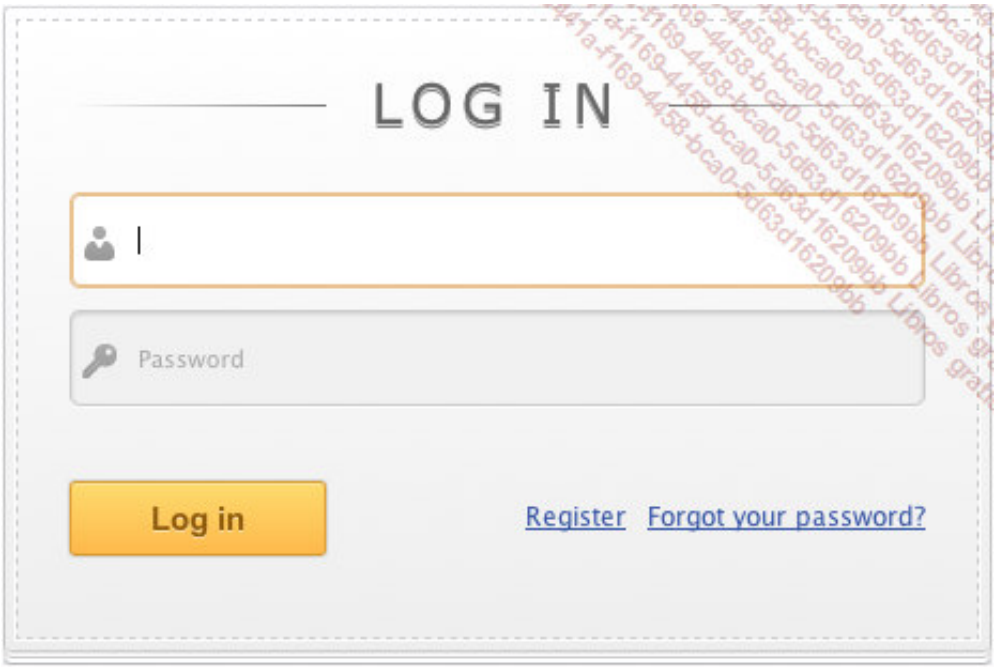
```
input[type=submit] {
    width: 185px;
    height: 52px;
    float: right;
    padding: 10px 15px;
    margin: 0 15px 0 0;
    -moz-box-shadow: 0px 0px 5px #999;
    -webkit-box-shadow: 0px 0px 5px #999;
    border: 1px solid #556f8c;
    background: -moz-linear-gradient(top, #718DA9 0%, #415D79
100%); /* firefox */
    background: -webkit-gradient(linear, left top, left bottom,
color-stop(0%,#718DA9), color-stop(100%,#415D79)); /* webkit */
    cursor: pointer;
}
```

Un segundo ejemplo de formulario muy sobrio

1. Objetivo

Vamos a ver otro ejemplo de un formulario muy sobrio, que utiliza al máximo el HTML5 y las hojas de estilo CSS3. Este formulario ha sido creado por el equipo de Red Team Design: http://www.red-team-design.com/slick-login-form-with-html5-css3

Y el resultado visual obtenido:

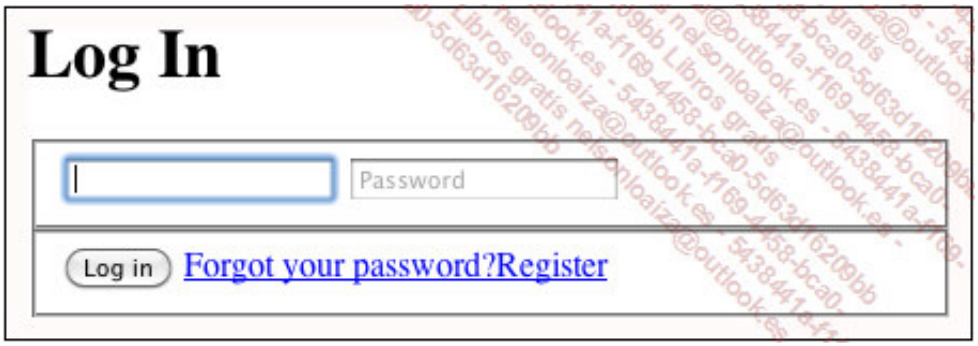


2. La estructura inicial del formulario

Se trata de un formulario clásico. Observe que se han usado los atributos `placeholder`,`autofocus` y `required`.

```
<form id="login">
    <h1>Log In</h1>
    <fieldset id="inputs">
        <input id="username" type="text"
placeholder="Username" autofocus required>
        <input id="password" type="password"
placeholder="Password" required>
    </fieldset>
    <fieldset id="actions">
        <input type="submit" id="submit" value="Log in">
        <a href="">Forgot your password?</a><a
href="">Register</a>
    </fieldset>
</form>
```

Así se visualiza sin estilos:



3. La sombra del formulario

Al campo de usuario del formulario se le ha aplicado un estilo sombreado mediante seis sombras:

```
#login {
    position: absolute;
    height: 240px;
    width: 400px;
    padding: 30px;
    -moz-box-shadow:
        0 0 2px rgba(0, 0, 0, 0.2),
        0 1px 1px rgba(0, 0, 0, .2),
        0 3px 0 #fff,
        0 4px 0 rgba(0, 0, 0, .2),
        0 6px 0 #fff,
        0 7px 0 rgba(0, 0, 0, .2);
    -webkit-box-shadow:
        0 0 2px rgba(0, 0, 0, 0.2),
        0 1px 1px rgba(0, 0, 0, .2),
        0 3px 0 #fff,
        0 4px 0 rgba(0, 0, 0, .2),
        0 6px 0 #fff,
        0 7px 0 rgba(0, 0, 0, .2);
    box-shadow:
        0 0 2px rgba(0, 0, 0, 0.2),
        0 1px 1px rgba(0, 0, 0, .2),
        0 3px 0 #fff,
        0 4px 0 rgba(0, 0, 0, .2),
        0 6px 0 #fff,
        0 7px 0 rgba(0, 0, 0, .2);
}
```

4. El efecto punteado

Podemos ver una línea de puntos alrededor del formulario. Ese efecto se ha conseguido con el pseudo-elemento `:before`. En principio, ese pseudo-elemento permite añadir contenido textual delante del elemento al que se aplique. En este ejemplo, los diseñadores lo han "adaptado" para añadir "espacios" (ausencia de texto) sobre el formulario. Por último, han aplicado un borde de guiones y una sombra ligera. ¡Qué buena idea!

```
#login:before {
    content: '';
    position: absolute;
    z-index: -1;
    border: 1px dashed #ccc;
    top: 5px;
    bottom: 5px;
```

```
    left: 5px;
    right: 5px;
    -moz-box-shadow: 0 0 0 1px #fff;
    -webkit-box-shadow: 0 0 0 1px #fff;
    box-shadow: 0 0 0 1px #fff;
}
```

La visualización:

5. El título del formulario

Al título del formulario se le ha aplicado una sombra, simplemente:

```
h1 {
    text-shadow: 0 1px 0 rgba(255, 255, 255, .7), 0px 2px 0
rgba(0, 0, 0, .5);
    text-transform: uppercase;
    text-align: center;
    color: #666;
    margin: 0 0 30px 0;
    letter-spacing: 4px;
    font: normal 26px/1 Verdana, Helvetica;
    position: relative;
}
```

6. Los trazos decorativos

A cada lado del título tenemos un trazo difuminado. ¡Se ha hecho todo en CSS3 y el resultado es realmente impresionante! Los diseñadores han vuelto a usar los pseudo-elementos `:before` y `:after` con un contenido vacío y luego han aplicado varios degradados lineales.

```
h1:after, h1:before {
    background-color: #777;
    content: "";
    height: 1px;
    position: absolute;
    top: 15px;
    width: 120px;
}

h1:after {
    background-image: -webkit-gradient(linear, left top, right
top, from(#777), to(#fff));
    background-image: -webkit-linear-gradient(left, #777, #fff);
```

```
    background-image: -moz-linear-gradient(left, #777, #fff);
    background-image: -ms-linear-gradient(left, #777, #fff);
    background-image: -o-linear-gradient(left, #777, #fff);
    background-image: linear-gradient(left, #777, #fff);
    right: 0;
}

h1:before {
    background-image: -webkit-gradient(linear, right top, left
top, from(#777), to(#fff));
    background-image: -webkit-linear-gradient(right, #777, #fff);
    background-image: -moz-linear-gradient(right, #777, #fff);
    background-image: -ms-linear-gradient(right, #777, #fff);
    background-image: -o-linear-gradient(right, #777, #fff);
    background-image: linear-gradient(right, #777, #fff);
    left: 0;
}
```

Este es el resultado visual visto de cerca:

7. Los campos de inserción de datos

Los campos de inserción de datos se han colocado en un elemento `fieldset`:

```
fieldset {
    border: 0;
    padding: 0;
    margin: 0;
}
```

Los campos de inserción `input` presentan un pequeño pictograma en su interior, se trata simplemente de una imagen no repetida que se ha usado para los dos campos. Los campos tienen esquinas redondeadas y una sombra.

```
#inputs input {
    background: #f1f1f1 url(http://www.red-team-design.com/
wp-content/uploads/2011/09/login-sprite.png) no-repeat;
    padding: 15px 15px 15px 30px;
    margin: 0 0 10px 0;
    width: 353px;
    border: 1px solid #ccc;
    -moz-border-radius: 5px;
    -webkit-border-radius: 5px;
    border-radius: 5px;
    -moz-box-shadow: 0 1px 1px #ccc inset, 0 1px 0 #fff;
    -webkit-box-shadow: 0 1px 1px #ccc inset, 0 1px 0 #fff;
    box-shadow: 0 1px 1px #ccc inset, 0 1px 0 #fff;
}
```

A continuación se ha posicionado correctamente la imagen de fondo para cada campo:

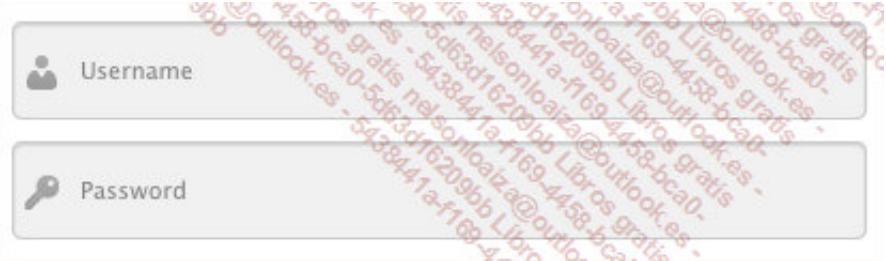
```
#username {
    background-position: 5px -2px !important;
}
#password {
    background-position: 5px -52px !important;
```

```
}
```

Cuando un campo de inserción de datos esté siendo utilizado, se le aplicará el estilo de "focus". La pseudo-clase `:focus` permite aplicar un diseño específico, en este caso, una sombra:

```
#inputs input:focus {
    background-color: #fff;
    border-color: #e8c291;
    outline: none;
    -moz-box-shadow: 0 0 0 1px #e8c291 inset;
    -webkit-box-shadow: 0 0 0 1px #e8c291 inset;
    box-shadow: 0 0 0 1px #e8c291 inset;
}
```

Así se visualizarán los campos cuando no se estén usando (sin *focus*):



El campo **Username** está ahora activo:

8. El botón de conexión

El botón de conexión **Log in** se encuentra en el recuadro `actions`.

```
#actions {
    margin: 25px 0 0 0;
}
```

El botón **Log in** utiliza el código de identificación `#submit`. Se le ha aplicado principalmente un degradado, un texto sombreado y una sombra de caja:

```
#submit {
    /* Diseño de la caja */
    border-width: 1px;
    border-style: solid;
    border-color: #d69e31 #e3a037 #d5982d #e3a037;
    float: left;
    height: 35px;
    padding: 0;
    width: 120px;
    cursor: pointer;
    font: bold 15px Arial, Helvetica;
    color: #8f5a0a;
```

```
    /* Degradado */
    background-color: #ffb94b;
    background-image: -webkit-gradient(linear, left top, left
bottom, from(#fddb6f), to(#ffb94b));
    background-image: -webkit-linear-gradient(top, #fddb6f,
#ffb94b);
    background-image: -moz-linear-gradient(top, #fddb6f, #ffb94b);
    background-image: -ms-linear-gradient(top, #fddb6f, #ffb94b);
    background-image: -o-linear-gradient(top, #fddb6f, #ffb94b);
    background-image: linear-gradient(top, #fddb6f, #ffb94b);
    /* Bordes redondeados */
    -moz-border-radius: 3px;
    -webkit-border-radius: 3px;
    border-radius: 3px;
    /* Sombra del texto */
    text-shadow: 0 1px 0 rgba(255,255,255,0.5);
    /* Sombra de la caja */
    -moz-box-shadow: 0 0 1px rgba(0, 0, 0, 0.3), 0 1px 0
rgba(255, 255, 255, 0.3) inset;
    -webkit-box-shadow: 0 0 1px rgba(0, 0, 0, 0.3), 0 1px 0
rgba(255, 255, 255, 0.3) inset;
    box-shadow: 0 0 1px rgba(0, 0, 0, 0.3), 0 1px 0 rgba(255,
255, 255, 0.3) inset;
}
```

Este es el botón resultante:

9. El uso del botón

Cuando se use el botón (`:focus`, `:hover` y `:active`), el degradado cambiará:

```
#submit:hover,#submit:focus {
    background-color: #fddb6f;
    background-image: -webkit-gradient(linear, left top, left
bottom, from(#ffb94b), to(#fddb6f));
    background-image: -webkit-linear-gradient(top, #ffb94b, #fddb6f);
    background-image: -moz-linear-gradient(top, #ffb94b, #fddb6f);
    background-image: -ms-linear-gradient(top, #ffb94b, #fddb6f);
    background-image: -o-linear-gradient(top, #ffb94b, #fddb6f);
    background-image: linear-gradient(top, #ffb94b, #fddb6f);
}
#submit:active {
    outline: none;
    -moz-box-shadow: 0 1px 4px rgba(0, 0, 0, 0.5) inset;
    -webkit-box-shadow: 0 1px 4px rgba(0, 0, 0, 0.5) inset;
    box-shadow: 0 1px 4px rgba(0, 0, 0, 0.5) inset;
}
#submit::-moz-focus-inner {
    border: none;
}
```

El resultado obtenido:

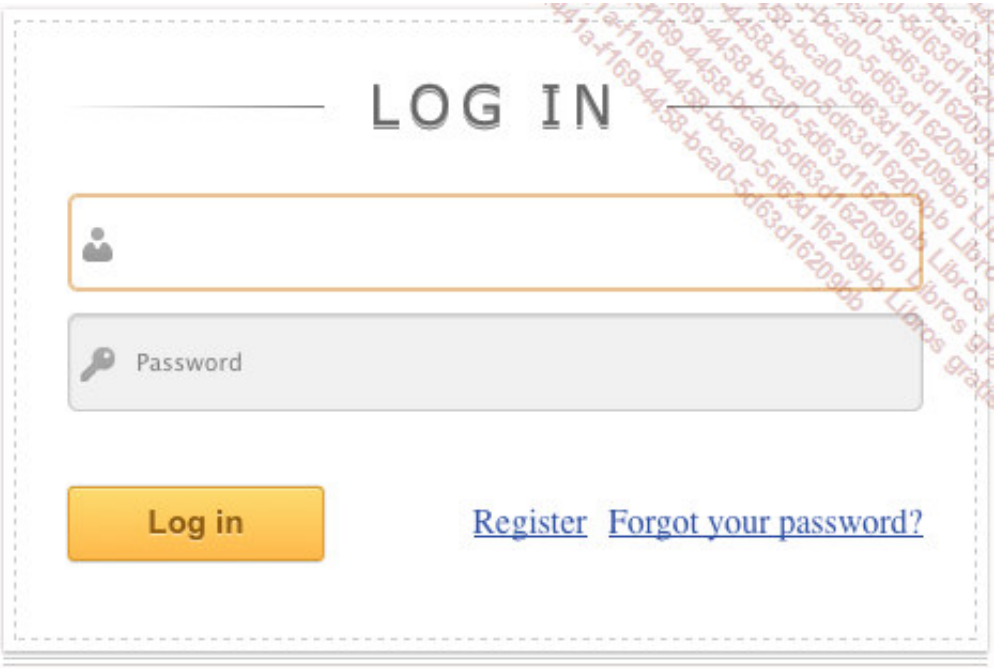
10. Los demás vínculos

El formato de los dos vínculos restantes es muy sencillo:

```
#actions a {
    color: #3151A2;
    float: right;
    line-height: 35px;
    margin-left: 10px;
}
```

11. El resultado final

Este es el formulario con la estructura en HTML5 y el diseño en CSS3:

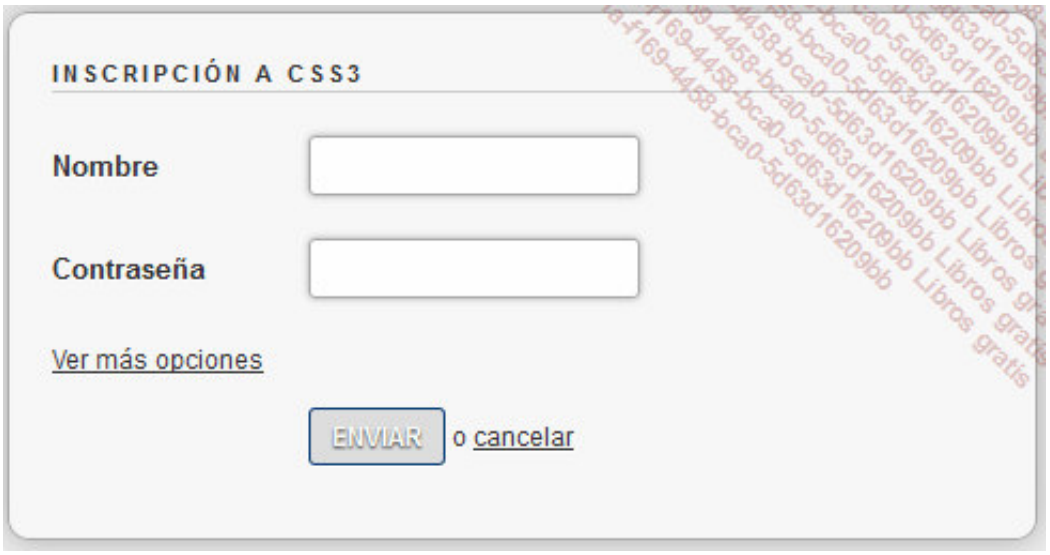


Un ejemplo de formulario interactivo

1. Objetivo

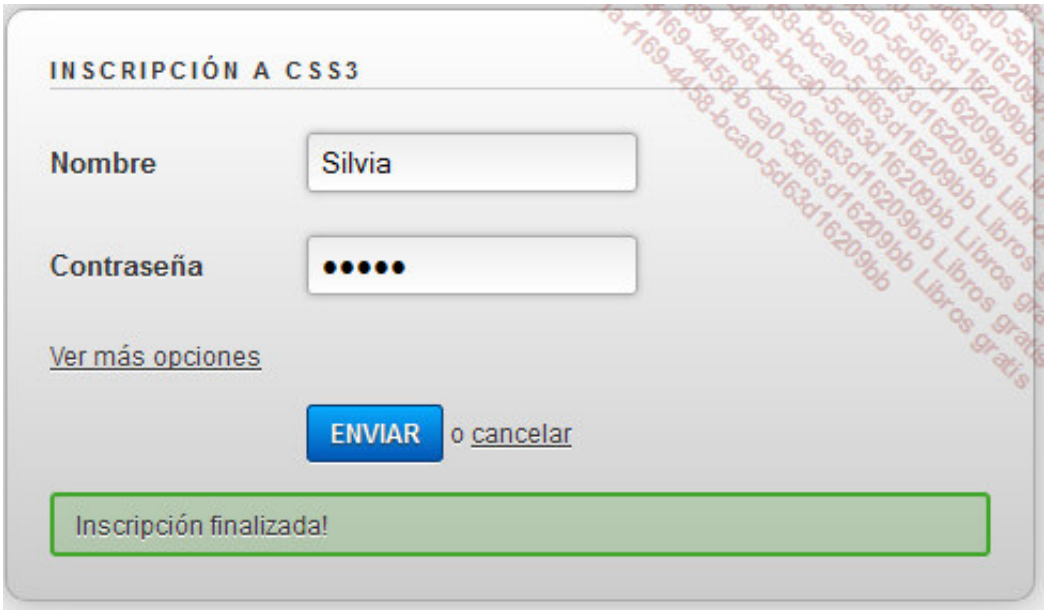
Vamos a explicar paso a paso la realización del formulario en CSS3.

Esta es la presentación visual del formulario:



Si se ha hecho clic en el vínculo que permite ver más opciones: **Ver más opciones**.

Cuando complete el formulario y haga clic en el botón **ENVIAR**, aparecerá un mensaje de confirmación:



2. La estructura inicial del formulario

La estructura es bastante clásica, con dos campos, uno para el **Nombre** y otro para la **Contraseña**:

```
<form id="start" action="/">
   <h1>Inscripción a CSS3</h1>
   <p>
      <label for="name">Nombre</label>
      <input id="name" type="text" />
   </p>
   <p>
      <label for="password">Contraseña</label>
      <input id="password" type="password" />
   </p>
</form>
```

3. Las opciones adicionales

Las opciones adicionales para el **Código postal**, la **Ciudad** y el **País**, se encuentran en una caja`<div id="options">`, para que sea posible gestionar la interactividad, justo debajo de los campos anteriores.

En esa misma caja `<div>` se encuentra el vínculo para acceder a las opciones adicionales **Ver más opciones**, en un párrafo `<p>` que utiliza la clase llamada `show`. Esta caja también contiene el vínculo para **Ocultar las opciones**, en un párrafo `<p>` que utiliza la clase llamada `hide`.

```
<div id="options">
    <p class="show">
        <a href="#options">Ver más opciones</a>
    </p>
    <p class="hide">
        <a href="#start">Ocultar las opciones</a>
    </p>
    <p>
        <label for="zipcode">Código postal</label>
        <input id="zipcode" type="text" />
    </p>
    <p>
        <label for="town">Ciudad</label>
```

```
        <input id="town" type="text" />
    </p>
    <p>
        <label for="country">País</label>
        <select id="country">
            <option>Francia</option>
            <option>Alemania</option>
            <option>Inglaterra</option>
            <option>España</option>
            <option>Italia</option>
            <option>Países Bajos</option>
            <option>Rusia</option>
            <option>Suecia</option>
        </select>
    </p>
</div>
```

4. El botón de envío

El botón de envío (`type="submit"`) ha sido insertado en un párrafo `<p>` en comentario, para que sea posible mostrar el mensaje de confirmación con los estilos CSS. Los dos elementos interactivos, el botón **ENVIAR** y el vínculo de cancelación **cancelar**, han sido insertados en vínculos:

```
<!--p>
    <input type="submit" value="ENVIAR" /> o <a
href="#form">cancelar</a>
</p-->
<p>
    <a class="submit" href="#finish">ENVIAR</a> o <a
href="#start">cancelar</a>
</p>
```

5. El mensaje de confirmación

Una vez que haya completado y enviado el formulario, aparecerá un mensaje de confirmación. Ese mensaje aparece dentro de una caja `<div id="finish">`.

```
<div id="finish">
    <p>
        Inscripción finalizada!
    </p>
</div>
```

6. Los estilos generales

En primer lugar, el diseñador web ha definido el estilo de un gran número de elementos HTML, pero no creemos que sea necesario repetir esa larga lista aquí. Si lo desea puede consultarla en la URL que indicamos anteriormente.

7. El diseño del formulario

El formulario (`form`) presenta un diseño con degradados, bordes redondeados y sombras, que se han adaptado para los principales navegadores.

```
form{
    background:#f7f7f7;
```

```
    background:-moz-linear-gradient(90deg, #ccc, #fff); /*
Firefox */
    background:-webkit-gradient(linear, left top, left bottom,
from(#fff), to(#ccc)); /* Webkit */
    border:1px solid #aaa;
    -moz-border-radius:10px;
    -webkit-border-radius:10px;
    -moz-box-shadow:0 0 15px #aaa;
    -webkit-box-shadow:0 0 15px #aaa;
    margin:60px auto 0;
    padding:20px;
    width:440px;
}
```

El título del formulario es un `h1`:

```
h1{
    border-bottom:1px solid #ccc;
    font-size:11px;
    font-weight:bold;
    letter-spacing:2px;
    margin-bottom:20px;
    text-transform:uppercase;
}
```

Los párrafos tienen un margen inferior de `0`. Fíjese en que se ha usado la pseudo-clase `:last-child` para seleccionar el último párrafo del formulario.

```
form p{
    margin-bottom:20px;
}
form p:last-child{
    margin-bottom:0;
}
```

8. Las etiquetas

Las etiquetas (`label`) presentan, entre otras propiedades, un cursor en forma de mano.

```
label{
    cursor:pointer;
    display:block;
    float:left;
    font-size:13px;
    font-weight:bold;
    line-height:28px;
    margin-bottom:5px;
    width:120px;
}
```

Las etiquetas cambian de apariencia cuando se pasa el cursor sobre ellas (`hover`): aparece el carácter **»** y el texto cambia de color.

La etiqueta sin actividad:

La misma etiqueta cuando se pasa el cursor sobre ella:

El selector `form p:hover label` se aplicará en un formulario (`form`) a un párrafo (`p`) cuando se señale con el cursor del ratón (`:hover`), siempre y cuando contenga la etiqueta (`label`):

```
form p:hover label{
    color:#0459b7;
}
```

Con la pseudo-clase `:after`, la hoja de estilo generará el contenido (`content`): **»** ("»").

```
form p:hover label:after{
    content:"»";
}
```

9. Los campos de inserción de datos

Los campos de inserción de texto (`input[type=text]`) y de contraseña (`input[type=password]`) presentan un diseño con fondos de color y transparencia, degradados y bordes redondeados.

```
input[type=text],
input[type=password]{
    background: rgba(255, 255, 255, 0.9);
    background:-moz-linear-gradient(90deg, #fff, #eee); /*
Firefox */
    background:-webkit-gradient(linear, left top, left bottom,
from(#eee), to(#fff), color-stop(0.2, #fff)); /* Webkit */
    border:1px solid #aaa;
    -moz-border-radius:3px;
    -webkit-border-radius:3px;
    -moz-box-shadow:0 0 3px #aaa;
    -webkit-box-shadow:0 0 3px #aaa;
    padding:5px;
}
```

Cuando uno de esos campos esté activo, es decir, presente el elemento `focus`, se modificará su formato con un borde azul y una sombra.

```
input[type=text]:focus,
input[type=password]:focus{
    border-color:#093c75;
    -moz-box-shadow:0 0 3px #0459b7;
    -webkit-box-shadow:0 0 3px #0459b7;
    outline:none;
}
```

10. El menú desplegable

El menú desplegable para los países (**País**) mostrará una sombra y un borde cuando se pase el cursor por encima.

```
select{
   cursor:pointer;
   padding:3px;
   -moz-box-shadow:0 0 3px #aaa;
   -webkit-box-shadow:0 0 3px #aaa;
}
select:active,
select:focus{
   border:1px solid #093c75;
   -moz-box-shadow:0 0 3px #0459b7;
   -webkit-box-shadow:0 0 3px #0459b7;
   outline:none;
}
```

11. El botón de envío

El botón para enviar el formulario (`input[type=submit]`) y el vínculo asociado (`a.submit`) presentan, como estilos CSS3 más destacados, un fondo degradado, bordes redondeados y una sombra aplicada a la caja y al texto.

```
input[type=submit],
a.submit{
    background:#ddd;
    background:-moz-linear-gradient(90deg, #0459b7, #08adff);
/* Firefox */
    background:-webkit-gradient(linear, left top, left bottom,
from(#08adff), to(#0459b7)); /* Webkit */
    border:1px solid #093c75;
    -moz-border-radius:3px;
    -webkit-border-radius:3px;
    -moz-box-shadow:0 1px 0 #fff;
    -webkit-box-shadow:0 1px 0 #fff;
    color:#fff;
    cursor:pointer;
    font-family:Arial,sans-serif;
    font-size:12px;
    font-weight:bold;
    margin-left:120px;
    padding:5px 10px;
    text-decoration:none;
    text-shadow:0 1px 1px #333;
    text-transform:uppercase;
}
```

Así se visualizará el botón para enviar el formulario:

Al pasar el cursor sobre ellos (`:hover`), esos elementos interactivos tendrán un diseño diferente.

```
input[type=submit]:hover,
a.submit:hover{
    background:#eee;
    background:-moz-linear-gradient(90deg, #067cd3, #0bcdff);
    background:-webkit-gradient(linear, left top, left bottom,
```

```
from(#0bcdff), to(#067cd3));
    border-color:#093c75;
    text-decoration:none;
}
```

Así se visualizará el botón de envió al pasar el ratón por encima:

Por último, esos elementos también mostrarán un diseño específico cuando se haga clic en ellos (pseudo-clase `:active`) y cuando estén activos (pseudo-clase `:focus`).

```
input[type=submit]:active,
input[type=submit]:focus,
a.submit:active,
a.submit:focus{
    background:#ccc;
    background:-moz-linear-gradient(90deg, #0bcdff, #067cd3);
    background:-webkit-gradient(linear, left top, left bottom,
from(#067cd3), to(#0bcdff));
    border-color:#093c75;
    outline:none;
}
```

Así se visualizará el botón de envío cuando se haga clic en él:

12. La gestión de las opciones

Las opciones del formulario se han insertado en una caja `<div id="content">`.

```
#options{
    margin-bottom:20px;
}
```

13. La visualización inicial

Al visualizar inicialmente el formulario, podemos ver el vínculo para acceder a las opciones (**Ver más opciones**).

```
#options .show{
   display:block;
}
```

Este vínculo usa la clase `.show`.

```
<p class="show">
    <a href="#options">Ver más opciones</a>
</p>
```

Al visualizar inicialmente el formulario, las opciones, que se encuentran dentro de párrafos (`<p>`), quedan ocultas gracias a la propiedad `display` con el valor `none`.

```
#options p{
    display:none;
}
```

Al igual que los elementos que usan la clase `.hide`.

```
#options .hide{
    display:none;
}
```

Eso es lo que ocurre con el párrafo que contiene el vínculo para ocultar las opciones (**Ocultar las opciones**).

```
<p class="hide">
    <a href="#start">Ocultar las opciones</a>
</p>
```

14. Visualizar las opciones

El vínculo para acceder a las opciones (**Ver más opciones**) se encuentra dentro de un párrafo <p>que usa la clase `show`.

```
<p class="show">
    <a href="#options">Ver más opciones</a>
</p>
```

Cuando el usuario haga clic en dicho vínculo, **Ver más opciones**, se estará haciendo una llamada a los elementos indicados en el `href` del vínculo: `href="#options"`. Con el uso de la pseudo-clase `:target`, los párrafos `p` de `#options` serán seleccionados y se mostrarán:

```
#options:target p{
    display:block;
}
```

Y los elementos de `#options` que tengan la clase `.show` quedarán ocultos, como es el caso del vínculo para acceder a las opciones (**Ver más opciones**):

```
#options:target .show{
    display:none;
}
```

Ahora podremos ver el vínculo que permite **Ocultar las opciones**, así como los campos adicionales.

15. Ocultar las opciones

Ahora el vínculo para **Ocultar las opciones** es visible y se encuentra dentro de un párrafo `<p>` que utiliza la clase `.hide`.

```
<p class="hide">
    <a href="#start">Ocultar las opciones</a>
</p>
```

Cuando el usuario haga clic en ese vínculo, se estará haciendo referencia al valor de `href`, es decir, `#start`, que es el código de identificación del formulario:

```
<form id="start" action="/">
    ...
</form>
```

De este modo volveremos a visualizar el formulario en su estado inicial.

16. El mensaje de confirmación

Cuando completemos el formulario, aparecerá un mensaje de confirmación. Ese mensaje de confirmación se encuentra dentro de un párrafo `<p>`, en una caja `<div id="finish">`.

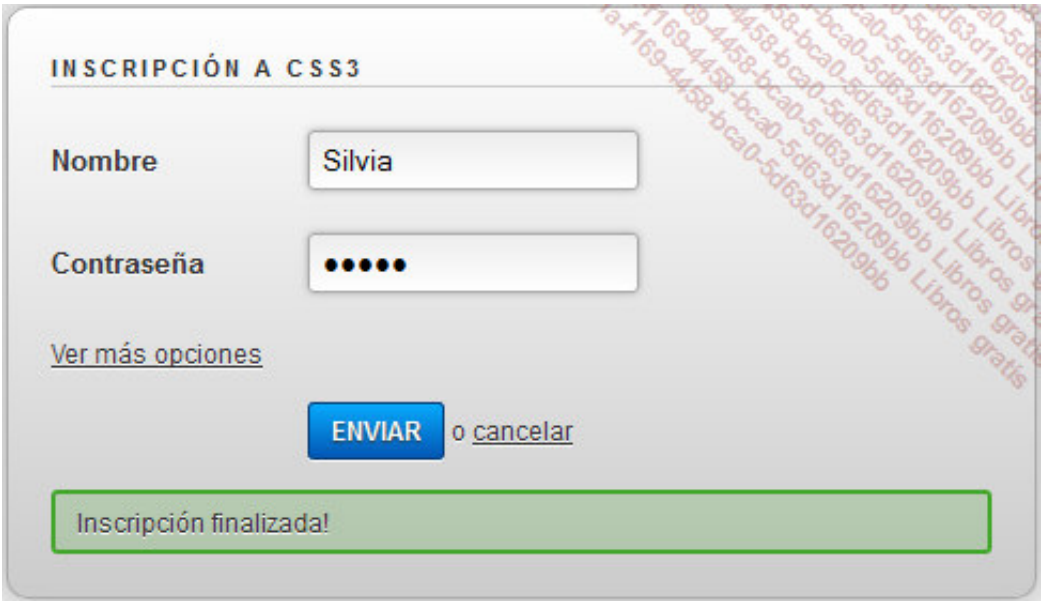
```
<div id="finish">
    <p>
        Inscripción finalizada!
    </p>
</div>
```

Inicialmente, esa caja `<div id="finish">` está oculta (`display: none`). Se le ha aplicado un fondo de color con transparencia (`rgba()`) y bordes redondeados.

```
#finish{
    background:rgba(65, 166, 42, 0.2);
    border:2px solid #41a62a;
    -moz-border-radius:3px;
    -webkit-border-radius:3px;
    display:none;
    padding:5px 10px;
}
```

Cuando el usuario haga clic en el botón de envío, la pseudo-clase `:target` permitirá aplicar esos estilos a la caja `<div id="finish">` y podremos visualizarla.

```
#finish:target {
    display:block;
}
```

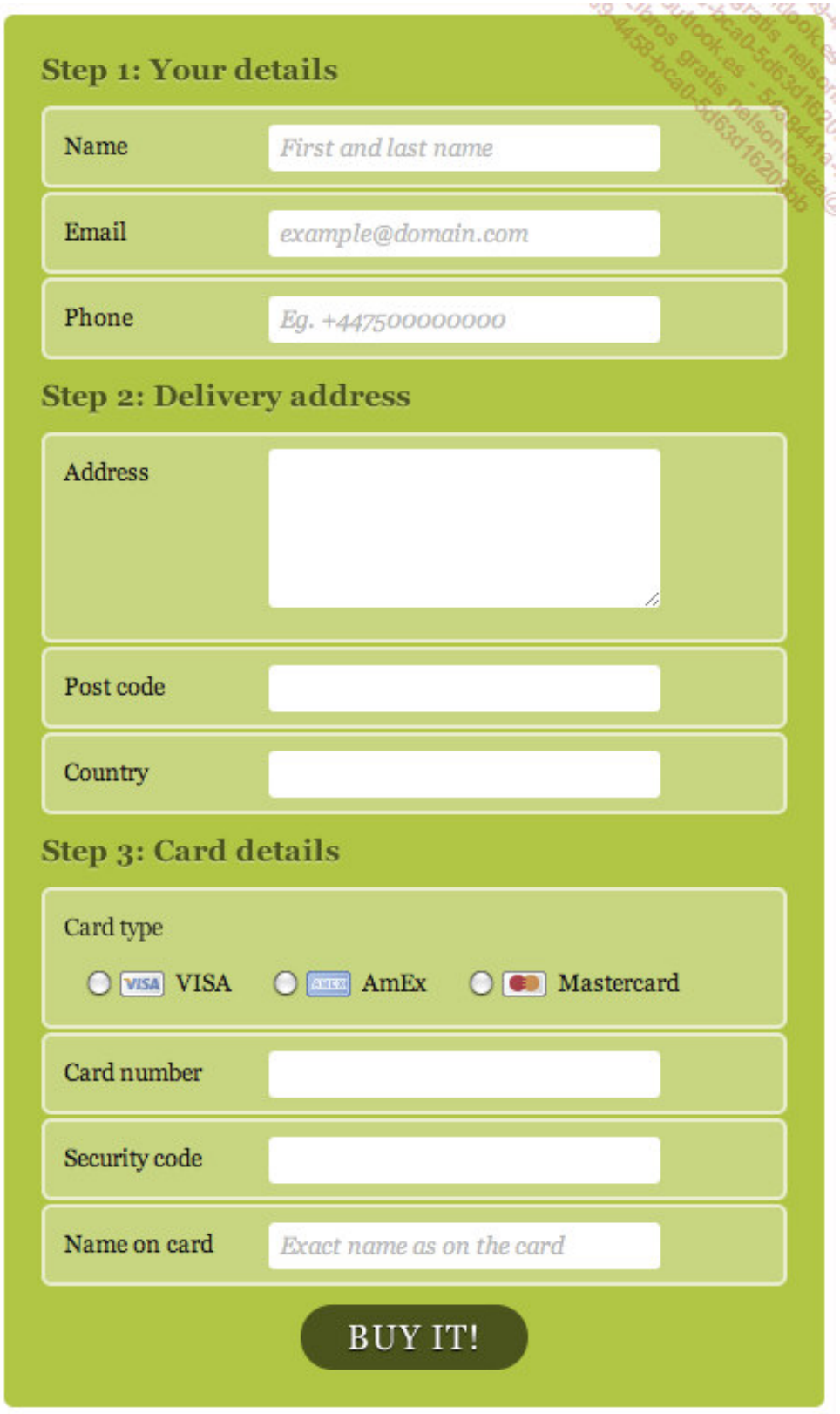


Un último ejemplo de formulario elaborado

1. Objetivo

Vamos a estudiar paso a paso la realización del formulario en HTML5/CSS3 que la diseñadora web Inayaili de León presenta en el sitio web: http://24ways.org/2009/have-a-field-day-with-html5-forms/

Este es el resultado final:



2. La estructura del formulario

Este formulario se compone de tres partes principales: **Your details**, **Delivery address** y **Card details**. Para empezar bien, con una estructura lógica y semántica, vamos a usar los elementos`<fieldset>` y `<legend>` para cada una de esas partes.

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Formulario de Inayaili de León</title>
    <meta charset="UTF-8" />
</head>
<body>
<form id=payment method=# action="#>
    <fieldset>
        <legend>Your details</legend>
    </fieldset>
    <fieldset>
        <legend>Delivery address</legend>
    </fieldset>
    <fieldset>
        <legend>Card details</legend>
    </fieldset>
    <fieldset>
        <button type=submit>Buy it!</button>
    </fieldset>
</form>
</body>
</html>
```

Como puede ver, la diseñadora ha preferido omitir las comillas en la declaración del valor de los atributos.

3. Los objetos HTML5

En HTML5 podemos usar nuevos atributos para los campos de texto, en función de su contenido:`email`, `number`, `tel`.

Además, podemos usar esos nuevos atributos para situar el punto de inserción en el primer campo (`autofocus`), para mostrar un ejemplo de un valor válido (`placeholder`) y para crear campos obligatorios (`required`).

La diseñadora ha insertado todos los objetos del formulario entre etiquetas `<ol>` y `<li>`. Efectivamente, es una buena idea, ya que un formulario se compone de una lista de objetos.

Veamos el código HTML de la sección **Your details**.

```
<fieldset>
    <legend>Your details</legend>
    <ol>
        <li>
           <label for=name>Name</label>
           <input id=name name=name type=text
placeholder="First and last name" required autofocus>
        </li>
        <li>
           <label for=email>Email</label>
           <input id=email name=email type=email
placeholder="example@domain.com" required>
        </li>
        <li>
           <label for=phone>Phone</label>
```

```
            <input id=phone name=phone type=tel
placeholder="Eg. +447500000000" required>
        </li>
    </ol>
</fieldset>
```

Así se visualizará la sección sin los estilos:

El primer campo de texto, **Name**, corresponde al nombre de usuario. Además de los atributos "tradicionales", este campo dispone de tres nuevos atributos HTML5:

- `placeholder="First and last name"`: permite incluir un valor de ejemplo en el campo.
- `required`: permite hacer que el campo sea obligatorio, que no pueda dejarse vacío.
- `autofocus`: permite hacer que el punto de inserción parpadee dentro de ese campo.

El segundo campo de texto, **Email**, es un campo de tipo e-mail: `type=email`. Esto permite añadir un control de los valores insertados. El campo deberá contener texto, seguido de arroba, y otra vez de texto, para que sea válido. Este campo también presenta los atributos `placeholder` y`required`.

El tercer campo de texto, **Phone**, es de tipo teléfono: `type=tel`. Recuerde que no hay ningún control de validación para los números de teléfono. Este campo también presenta los atributos`placeholder` y `required`.

En la sección **Card details**, encontramos dos campos de tipo `number` para introducir el número de la tarjeta de crédito y el código de seguridad.

```
<li>
    <label for=cardnumber>Card number</label>
    <input id=cardnumber name=cardnumber type=number required>
</li>
<li>
    <label for=secure>Security code</label>
    <input id=secure name=secure type=number required>
</li>
```

Usted puede usar una expresión regular con el atributo `pattern`, si desea exigir que se inserte un valor determinado.

4. El código completo del formulario

Este es el código completo del formulario:

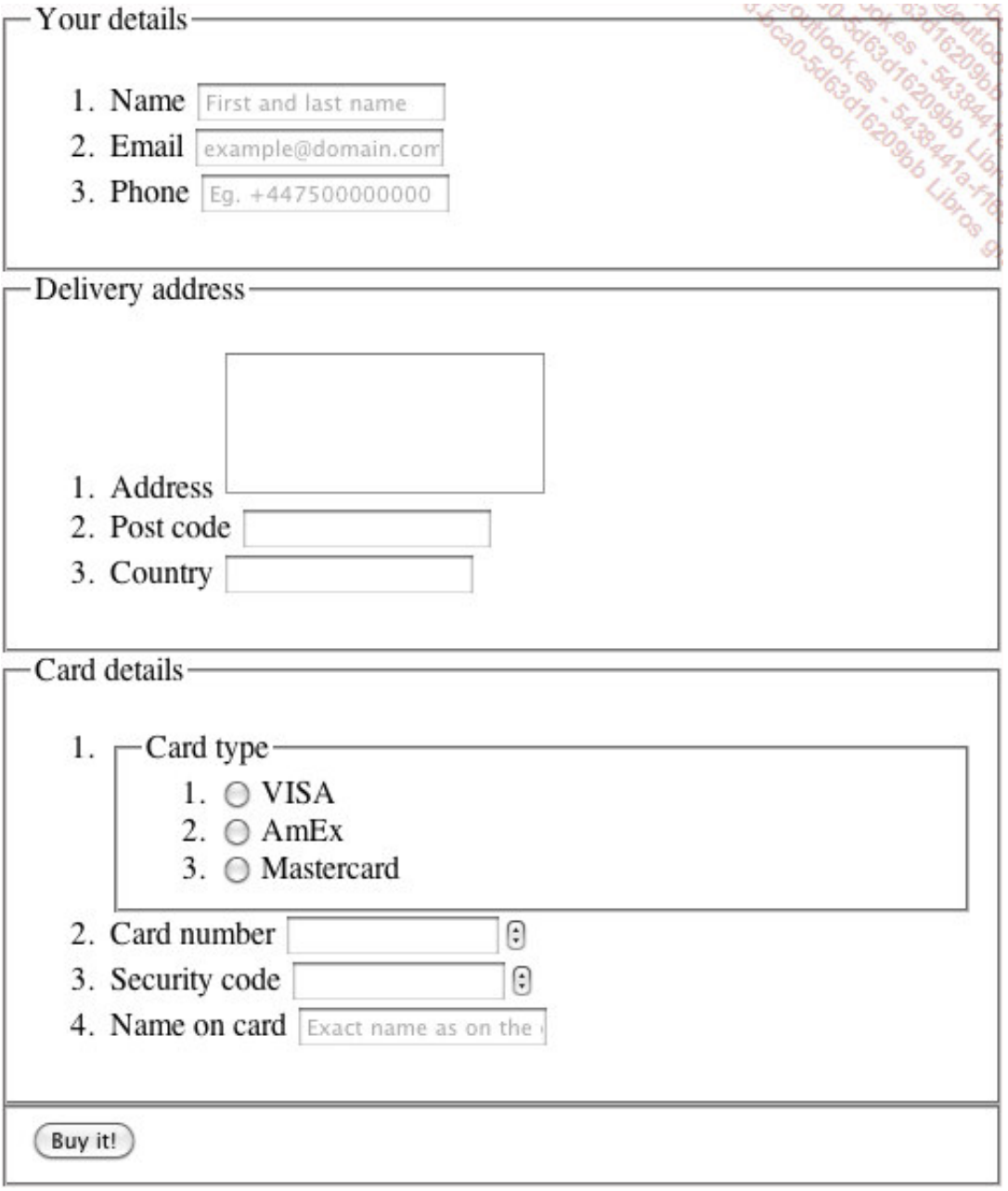
```
<form id=payment method=# action=#>
<fieldset>
    <legend>Your details</legend>
    <ol>
       <li>
```

```
            <label for=name>Name</label>
            <input id=name name=name type=text
placeholder="First and last name" required autofocus>
        </li>
        <li>
            <label for=email>Email</label>
            <input id=email name=email type=email
placeholder="example@domain.com" required>
        </li>
        <li>
            <label for=phone>Phone</label>
            <input id=phone name=phone type=tel
placeholder="Eg. +447500000000" required>
        </li>
    </ol>
</fieldset>
<fieldset>
    <legend>Delivery address</legend>
    <ol>
        <li>
            <label for=address>Address</label>
            <textarea id=address name=address rows=5
required></textarea>
        </li>
        <li>
            <label for=postcode>Post code</label>
            <input id=postcode name=postcode type=text required>
        </li>
        <li>
            <label for=country>Country</label>
            <input id=country name=country type=text required>
        </li>
    </ol>
</fieldset>
<fieldset>
    <legend>Card details</legend>
    <ol>
        <li>
           <fieldset>
               <legend>Card type</legend>
               <ol>
                   <li>
                      <input id=visa name=cardtype type=radio>
                      <label for=visa>VISA</label>
                   </li>
                   <li>
                      <input id=amex name=cardtype type=radio>
                      <label for=amex>AmEx</label>
                   </li>
                   <li>
                      <input id=mastercard name=cardtype type=radio>
                      <label for=mastercard>Mastercard</label>
                   </li>
               </ol>
           </fieldset>
        </li>
        <li>
           <label for=cardnumber>Card number</label>
           <input id=cardnumber name=cardnumber type=number required>
        </li>
        <li>
           <label for=secure>Security code</label>
           <input id=secure name=secure type=number required>
        </li>
        <li>
```

```
            <label for=namecard>Name on card</label>
            <input id=namecard name=namecard type=text
placeholder="Exact name as on the card" required>
        </li>
    </ol>
</fieldset>
<fieldset>
    <button type=submit>Buy it!</button>
</fieldset>
</form>
```

5. La visualización sin estilos

Esta es la apariencia que tendrá el formulario sin los estilos:



6. El diseño general

En primer lugar, la diseñadora ha definido los estilos de los elementos que componen la página mediante un margen y un espaciado de 0.

```
html, body, h1, form, fieldset, legend, ol, li {
    margin: 0;
    padding: 0;
 }
```

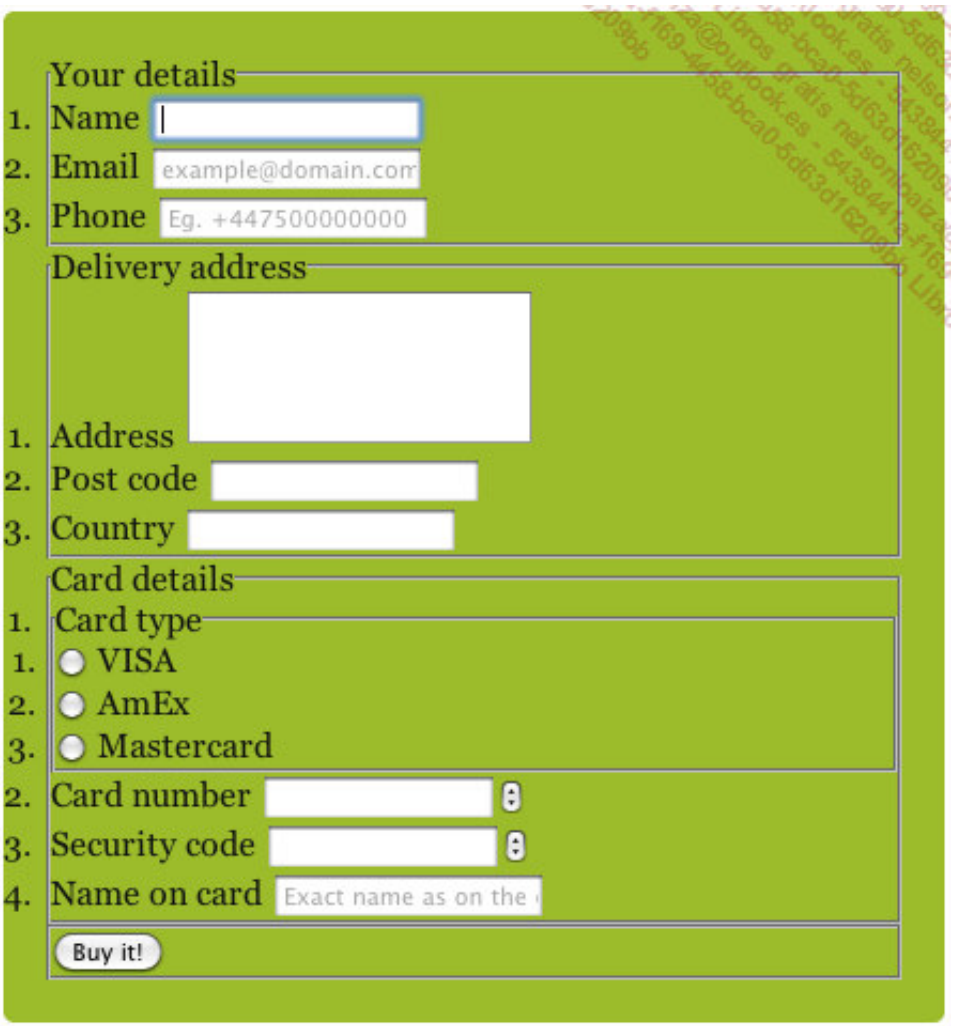
A continuación ha definido el formato del texto y del cuerpo de la página:

```
body {
    background: #ffffff;
    color: #111111;
    font-family: Georgia, "Times New Roman", Times, serif;
    padding: 20px;
 }
```

7. El diseño del formulario

El formulario presenta un diseño con un color de fondo, bordes redondeados, un espaciado y un ancho fijo.

```
form#payment {
    background: #9cbc2c;
    -moz-border-radius: 5px;
    -webkit-border-radius: 5px;
    border-radius: 5px;
    padding: 20px;
    width: 400px;
 }
```



8. El formato de las secciones

Cada sección del formulario utiliza los elementos `<fieldset>` y `<legend>`. Se les ha aplicado un diseño sin bordes y un margen inferior. Fíjese en que se ha usado la pseudo-clase `last-of-type`para seleccionar el último `<fieldset>`.

```
form#payment fieldset {
    border: none;
    margin-bottom: 10px;
}
form#payment fieldset:last-of-type {
    margin-bottom: 0;
}
```

El elemento `<legend>` de los elementos `<fieldset>` también aparece resaltado, en particular, con una sombra:

```
form#payment legend {
    color: #384313;
    font-size: 16px;
    font-weight: bold;
    padding-bottom: 10px;
    text-shadow: 0 1px 1px #c0d576;
}
```

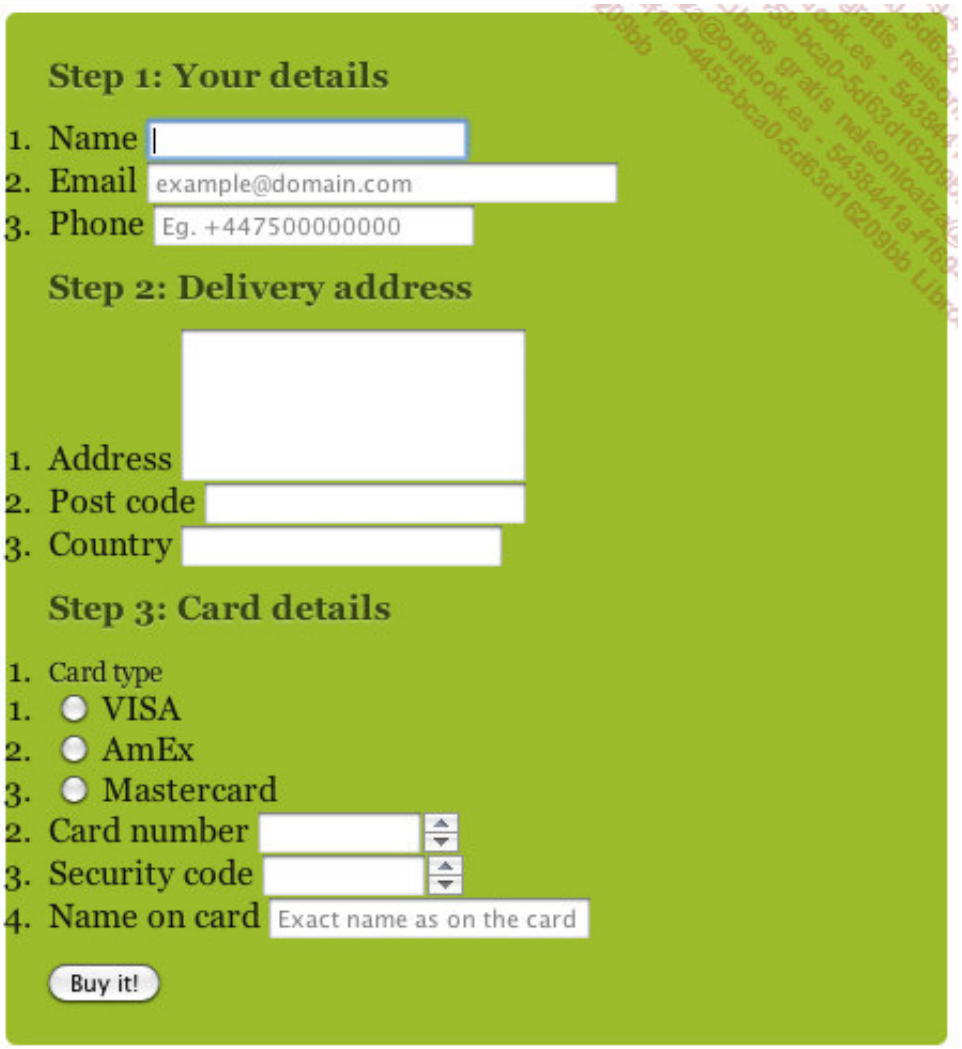
En cada sección del formulario, cada etiqueta `<legend>` aparece precedida de la palabra **Step**seguida de un número incrementado. Con el pseudo-elemento `:before` y las propiedades`content` y `counter-increment`, podemos automatizar esta enumeración.

```
    form#payment > fieldset > legend:before {
        content: "Step " counter(fieldsets) ": ";
    counter-increment: fieldsets;
    }
```

Por último, para el recuadro del tipo de tarjeta de crédito, la diseñadora web ha preferido crear un aspecto de tipo `<label>` y no `<legend>`.

```
form#payment fieldset fieldset legend {
    color: #111111;
    font-size: 13px;
    font-weight: normal;
    padding-bottom: 0;
}
```

Así se visualiza el formulario:



9. El diseño de los elementos de la lista

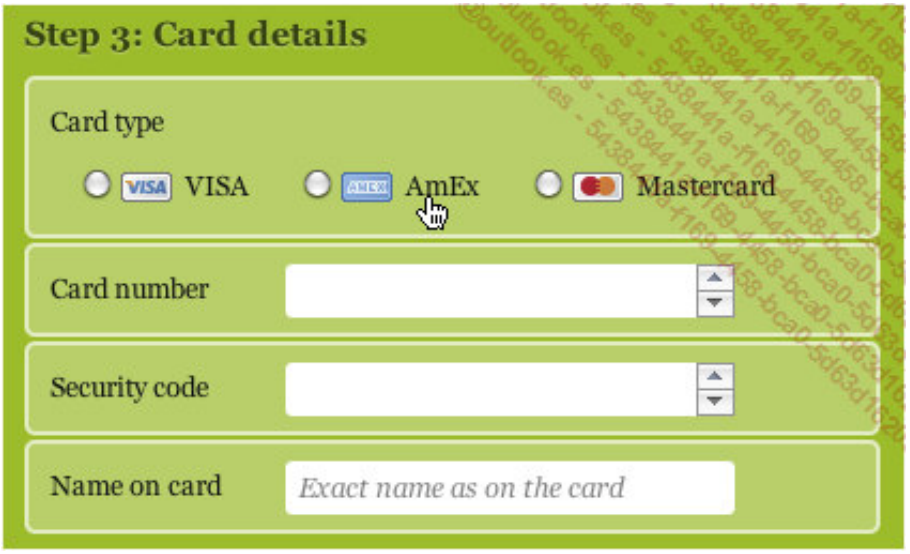
Todos los objetos del formulario están insertados en elementos `<li>` de una lista `<ol>`. Estos presentan un diseño con bordes redondeados y una ligera transparencia (`rgba()`).

```
form#payment ol li {
    background: #b9cf6a;
    background: rgba(255,255,255,.3);
    border-color: #e3ebc3;
    border-color: rgba(255,255,255,.6);
    border-style: solid;
    border-width: 2px;
    -moz-border-radius: 5px;
    -webkit-border-radius: 5px;
    border-radius: 5px;
    line-height: 30px;
    list-style: none;
    padding: 5px 10px;
    margin-bottom: 2px;
}
form#payment ol ol li {
    background: none;
    border: none;
    float: left;
}
```

10. El diseño de las etiquetas

A continuación se ha definido el diseño de las etiquetas de los campos `<label>` y, en especial, las de las tarjetas de crédito (`label[for=amex]`), que están ilustradas con una imagen y el cursor toma la forma de mano (`cursor: pointer`) cuando se pasa el ratón por encima de ellas (`:hover`).

```
form#payment label {
    float: left;
    font-size: 13px;
    width: 110px;
}
form#payment fieldset fieldset label {
    background:none no-repeat left 50%;
    line-height: 20px;
    padding: 0 0 0 30px;
    width: auto;
}
form#payment label[for=visa] {
    background-image: url(visa.gif);
}
form#payment label[for=amex] {
    background-image: url(amex.gif);
}
form#payment label[for=mastercard] {
    background-image: url(mastercard.gif);
}
form#payment fieldset fieldset label:hover {
    cursor: pointer;
}
```



11. Los campos de tipo <input>

Todos los campos de tipo `<input>` serán idénticos, a excepción de los botones de opción y el botón de validación. Para ello, fíjese en que se ha usado la pseudo-clase `:not()`: `input:not([type=radio])`.

```
form#payment input:not([type=radio]),
form#payment textarea {
    background: #ffffff;
    border: none;
    -moz-border-radius: 3px;
    -webkit-border-radius: 3px;
    -khtml-border-radius: 3px;
    border-radius: 3px;
    font: italic 13px Georgia, "Times New Roman", Times, serif;
    outline: none;
    padding: 5px;
    width: 200px;
}
form#payment input:not([type=submit]):focus,
form#payment textarea:focus {
    background: #eaeaea;
}
form#payment input[type=radio] {
float: left;
    margin-right: 5px;
}
```

12. El botón de envío del formulario

El botón de envío del formulario tiene un estilo propio.

```
form#payment button {
    background: #384313;
    border: none;
    -moz-border-radius: 20px;
    -webkit-border-radius: 20px;
    -khtml-border-radius: 20px;
    border-radius: 20px;
    color: #ffffff;
    display: block;
```

```
    font: 18px Georgia, "Times New Roman", Times, serif;
    letter-spacing: 1px;
    margin: auto;
    padding: 7px 25px;
    text-shadow: 0 1px 1px #000000;
    text-transform: uppercase;
}
form#payment button:hover {
    background: #1e2506;
    cursor: pointer;
}
```

Crear botones con símbolos

1. Objetivo

En sus formularios, puede que alguna vez necesite crear botones con símbolos o con pictogramas. Veamos cómo podemos crearlos fácilmente con las pseudo-clases `:before` y `:after`. Los ejemplos están adaptados del sitio web: http://www.paulund.co.uk/css-buttons-with-icons-but-no-images

2. El formulario

Vamos a crear un formulario clásico con dos botones, uno para validar el formulario y otro para anularlo. Sin embargo, no vamos a usar botones, sino vínculos, ya que la técnica que vamos a usar, con pseudo-elementos, no se puede aplicar a los botones.

Este es el formulario inicial:

```
<form id="registro" method="#" action="#">
    <p>
       <label for="nombre">Nombre: </label>
       <input type="nombre" name="nombre" id="nombre" />
    </p>
    <p>
       <label for="apellidos">Apellidos: </label>
       <input type="apellidos" name="apellidos" id="apellidos" />
    </p>
    <p>
       <a href="#">Confirme<a/>
       <a href="#">Anule<a/>
    </p>
</form>
```

Así se visualizará sin estilos:

3. Crear una clase para los botones

La técnica consiste en crear una clase para todos los botones que vaya a usar en su formulario. Esta clase permitirá diseñar el botón con todos los elementos de formato habituales. Podemos usar las propiedades CSS porque los vínculos aparecen como bloque en línea:`display: inline-block`.

Con esta clase aplicaremos a los botones un ligero degradado y bordes redondeados.

```
.botones {
    /* La caja*/
    display: inline-block;
    border: 1px solid #000;
```

```
    padding: .2em .5em;
    margin: 0 4em 0 0;
    /* El degradado */
    background: -moz-linear-gradient(top, #fff 0%, #eee 100%);
    background: -webkit-linear-gradient(top, #fff 0%, #eee 100%);
    background: -o-linear-gradient(top, #fff 0%,#ccc 100%);
    background: linear-gradient(top, #fff 0%,#ccc 100%);
    /* Los bordes redondeados */
    -moz-border-radius: .3em;
    -webkit-border-radius: .3em;
    border-radius: .3em;
    /* El texto */
    font-weight: bold;
    font-size: 1em;
    text-decoration: none;
    color: black;
}
```

Así se visualizará el formulario con los estilos:

4. El pseudo-elemento ::before

El pseudo-elemento `::before` (fíjese en la nueva sintaxis de los pseudo-elementos con los dos caracteres :) permite añadir texto delante del elemento al que esté asociado. El pseudo-elemento `::after` hace lo mismo, pero detrás de dicho elemento.

Tenga en cuenta que puede usar la sintaxis `:before` y `:after` si encuentra algún problema de compatibilidad con su navegador.

Vamos a aplicar el pseudo-elemento `::before` a la clase genérica de los botones `.botones`, para que el estilo sea más sencillo.

```
.botones:before {
    font-size: 1em;
    padding: 0 .5em 0 0;
}
```

5. El contenido del pseudo-elemento ::before

Ahora tendremos que indicar el contenido que vamos a añadir delante de la etiqueta del botón, el texto del vínculo `a`. Debemos especificar a qué elemento se le añadirá ese contenido, ya que cada botón tendrá un contenido diferente.

En nuestro ejemplo, vamos a crear dos selectores específicos `.validar:before` y `.anular:before`, con un estilo específico para cada botón.

Segundo parámetro: se trata del pictograma que vamos a insertar. La norma Unicode UTF-8 nos propone cientos de símbolos y de pictogramas. No deje de visitar estas dos URL: http://unicode-table.com/en y http://www.utf8-chartable.de

Sepa que también puede usar las entidades de caracteres HTML clásicas:http://www.w3.org/TR/html4/sgml/entities.html

Para la validación del formulario, queremos insertar el "símbolo de validación", que tiene el código de visualización `\2714`.

Para la anulación del formulario, queremos usar el símbolo de la "cruz de eliminación", que tiene el código de visualización `\2717`.

Estos son los dos estilos:

```
.validar:before {
    content: "\2714";
}
.anular:before {
    content: "\2717";
}
```

6. Aplicar las clases

Última etapa, ahora vamos a aplicar las clases que acabamos de crear a los elementos `a` de los vínculos del formulario. Los dos vínculos utilizan la clase genérica de los botones `botoncito` y cada uno de ellos utilizará la clase que le corresponda para la inserción de contenido: `validar` y`anular`.

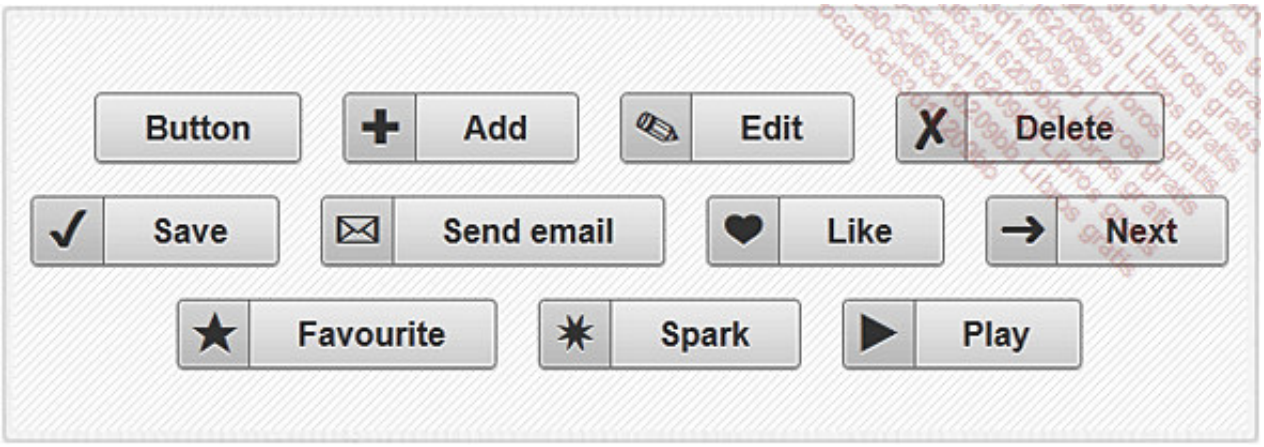
```
<a href="#" class="botoncito validar">Confirme<a/>
<a href="#" class="botoncito anular">Anule<a/>
```

7. La visualización del formulario

Así se visualizará el formulario:

8. Otros ejemplos similares

Encontrará otras variantes de esta técnica en la URL: http://www.red-team-design.com/just-another-awesome-css3-buttons


Button
Add
Edit
Delete
Save
Send email
Like
Next
Favourite
Spark
Play

Los generadores de botones en línea

1. Funcionalidad

Los estilos CSS3 nos aportan mejoras reales en cuanto a la creación de botones, hasta tal punto que ya no es necesario usar programas de diseño para crearlos. Los botones con imágenes solo presentan inconvenientes: aumentan considerablemente el tamaño de los archivos, las actualizaciones del diseño son más lentas, afectan negativamente al posicionamiento natural y a la accesibilidad.

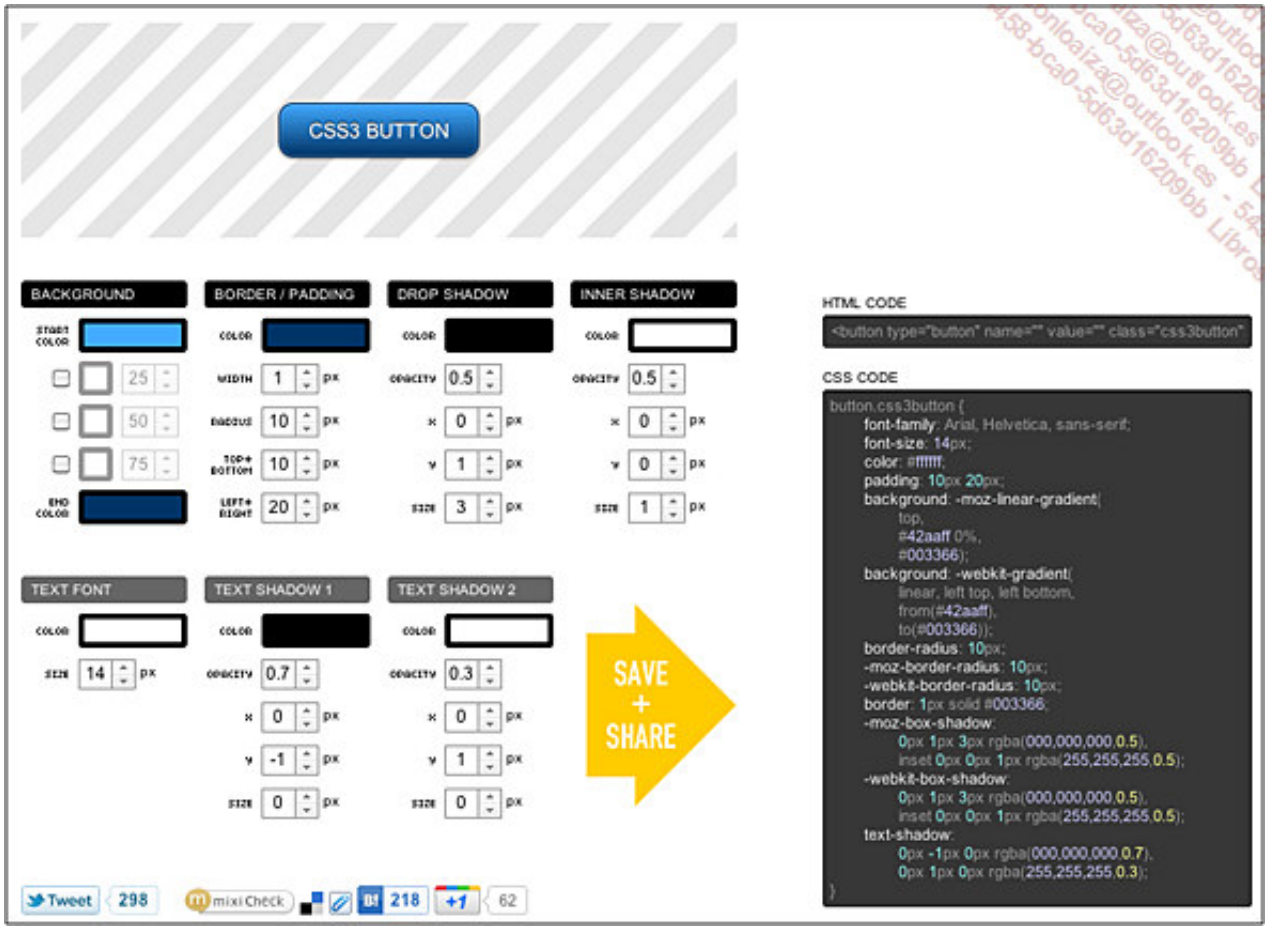
Para crear botones con un diseño moderno rápidamente, la web pone a su disposición multitud de generadores en línea. Veamos algunos de ellos.

2. CSS3 Button

Este sitio web (http://css3button.net/) tiene una interfaz agradable en la que podrá trabajar con:

- el degradado del fondo,
- los bordes y las esquinas redondeadas,
- el espaciamiento,
- las sombras externas e internas,
- el color del texto,
- las sombras de texto.

La visualización se actualizará automáticamente y podrá recuperar el código en la zona **CSS CODE**.

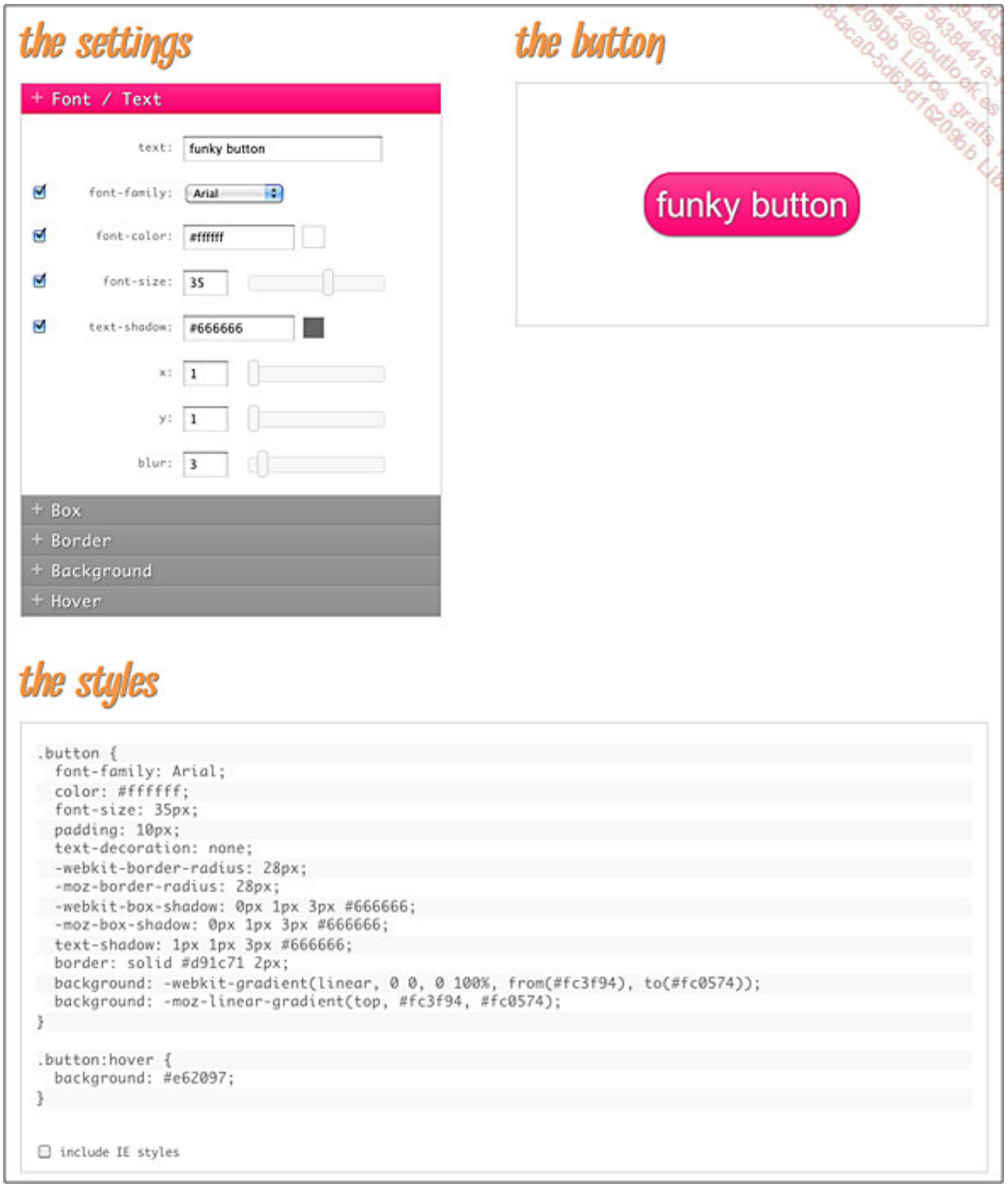


3. CSS3 Button Generator

Este sitio web (http://css3buttongenerator.com/) presenta una interfaz en acordeón en la que podrá seleccionar las propiedades CSS que desee aplicar:

- el texto sombreado,
- el espaciado y las sombras,
- los bordes y las esquinas redondeadas,
- el degradado del fondo,
- el degradado del fondo para `:hover`.

Como es habitual, podrá ver el resultado en tiempo real, así como el código CSS generado.



4. CSS Drive Button Generator

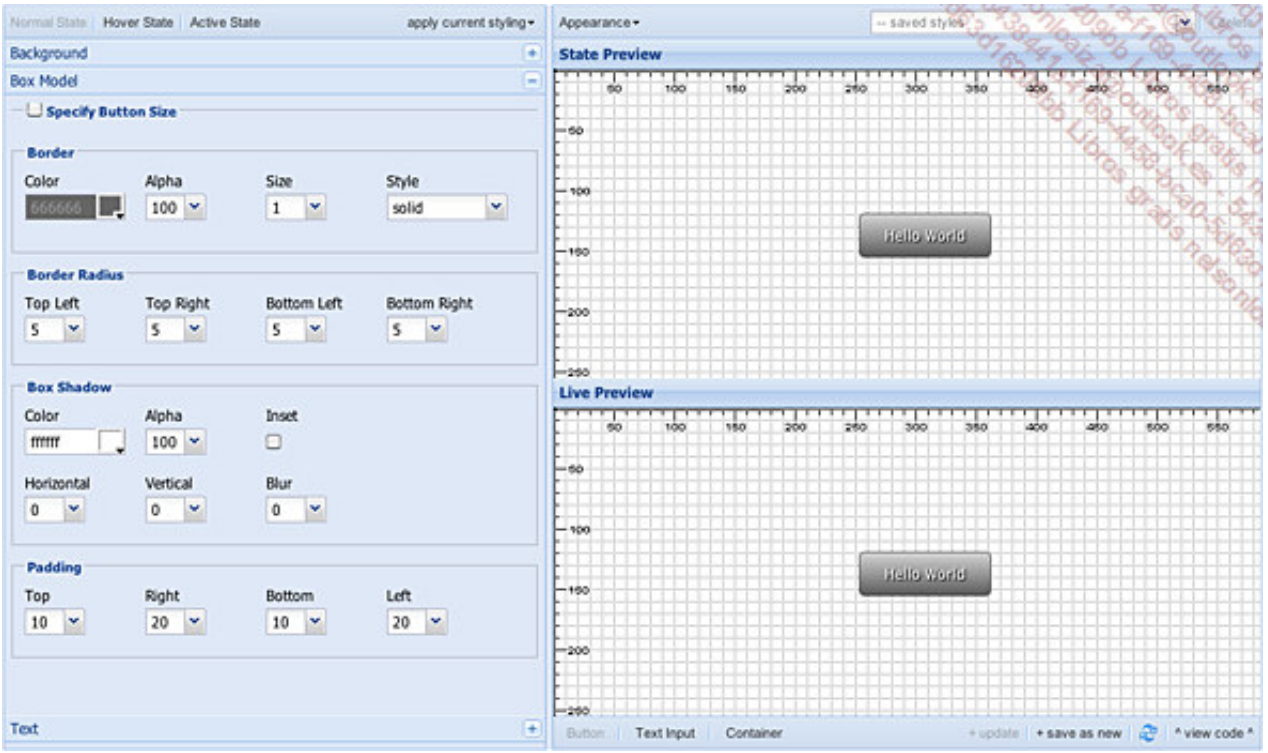
Este sitio web (http://www.cssdrive.com/css3button/) le propone crear botones de una forma muy completa: Podrá trabajar con:

- todos los parámetros del texto,
- la caja del botón: degradado para el fondo, ancho, espaciamiento...
- los bordes, las esquinas redondeadas y la sombra,
- y como colofón, las transformaciones, las transiciones y los estilos para `:hover`.

Nuevamente, podrá ver el resultado en tiempo real, así como el código CSS generado.

5. CSS3 Button Creator

El sitio web (http://www.cssbuttongenerator.com/) propone una interfaz innovadora y muy interactiva, en la que tendremos la impresión de estar usando realmente una aplicación. Podrá trabajar con todas las propiedades disponibles, a las que podrá acceder a partir de los paneles de la izquierda.


Normal State
Hover State
Active State
apply current styling
Background
Box Model
Specify Button Size
Border
Color
666666
Alpha
100
Size
1
Style
solid
Border Radius
Top Left
5
Top Right
5
Bottom Left
5
Bottom Right
5
Box Shadow
Color
ffffff
Alpha
100
Inset
Horizontal
0
Vertical
0
Blur
0
Padding
Top
10
Right
20
Bottom
10
Left
20
Text
Appearance
-- saved styles --
State Preview
Hello World
Live Preview
Hello World
Button
Text Input
Container
+ update
+ save as new
^ view code ^

Situación actual y objetivos

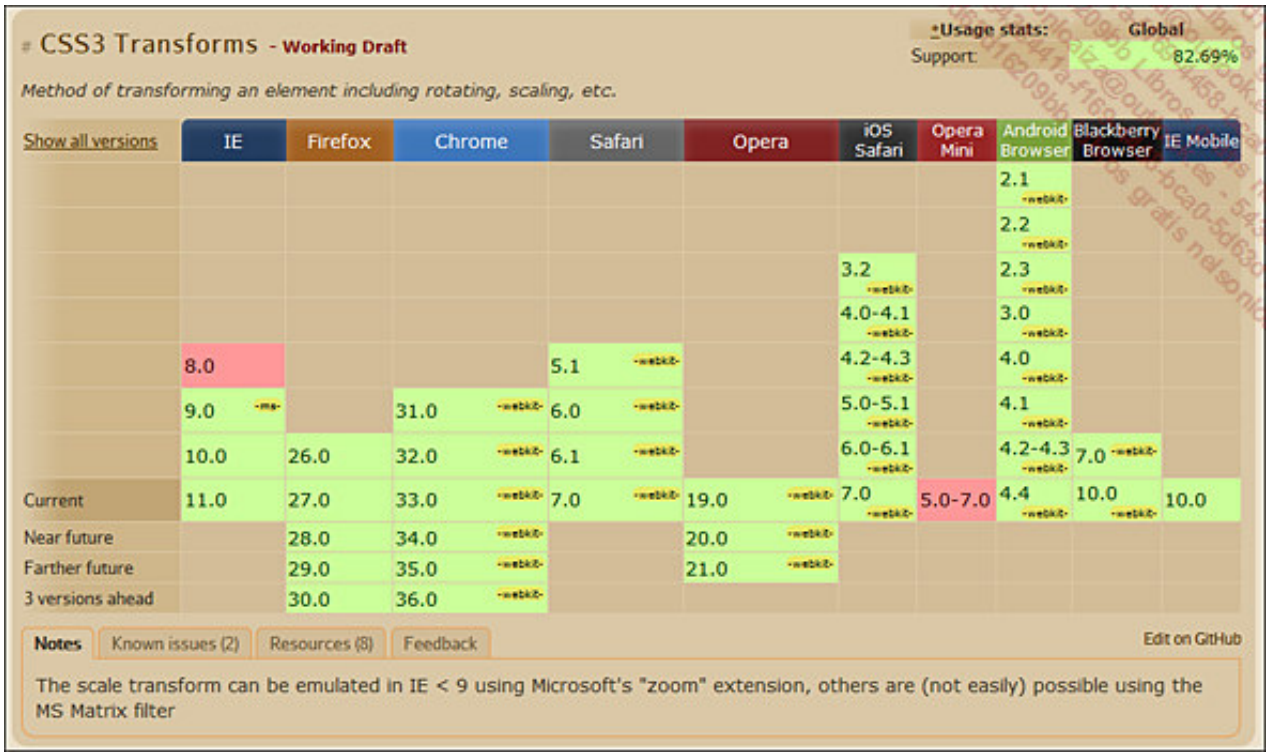
El módulo de las transformaciones fue desarrollado en un primer momento por los programadores del motor de visualización WebKit (de Safari y Chrome).

En la actualidad (septiembre de 2013), el módulo de transformaciones se encuentra aún en estado de borrador (**Working Draft**) y data del 11 de septiembre de 2012: http://www.w3.org/TR/css3-transforms/. Anteriormente existían dos módulos, uno para el 2D y otro para el 3D. Estos dos módulos se han fusionado en uno solo.

Las transformaciones permiten modificar la visualización de los elementos HTML de una página web con las propiedades CSS.

Una vez que el elemento HTML haya sido publicado de manera "tradicional", podrá "soportar" transformaciones de rotación, de desplazamiento, de deformación, de escala y de perspectiva.

Una vez más, los prefijos propietarios de los navegadores serán indispensables.



La transformación

1. La propiedad

Para aplicar una transformación tendremos que utilizar la propiedad `transform`. Esta propiedad utilizará a su vez diversas funciones dependiendo de la transformación en cuestión.

2. El punto de referencia

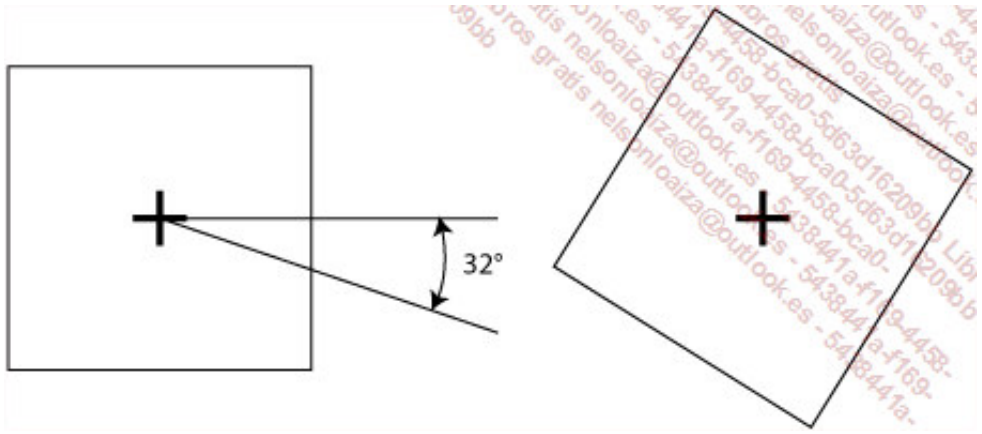
De manera predeterminada, todas las transformaciones tomarán como punto de referencia el centro del elemento. Ese punto de referencia sirve de valor inicial en los cálculos de las transformaciones.

Nosotros podemos cambiar ese punto de referencia con la propiedad `transform-origin`. El valor insertado indicará el nuevo punto de referencia.

Los valores posibles son:

- porcentajes: por defecto, el valor es de `50% 50%`, es decir, el centro del elemento.
- palabras clave: `left`, `center`, `right`, `top`, `center`, `bottom`.
- valores expresados en píxeles.

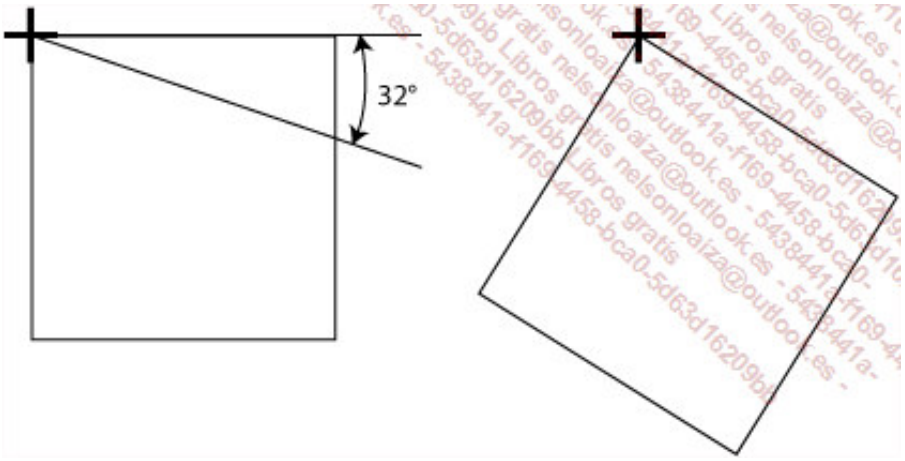
Este sería el resultado de una rotación con el punto de referencia predeterminado, es decir, en el centro del elemento:



Veamos un ejemplo de modificación del punto de origen a la esquina superior izquierda del elemento, seguido de una rotación:

```
.rotar {
    -moz-transform-origin: left top;
    -webkit-transform-origin: left top;
    -o-transform-origin: left top;
    transform-origin: left top;
    -moz-transform: rotate(32deg);
    -webkit-transform: rotate(32deg);
    -o-transform: rotate(32deg);
    transform: rotate(32deg);
}
```

Este sería el resultado obtenido al modificar del punto de referencia:


32°

El desplazamiento

1. En los dos ejes

La función `translate` permite realizar un desplazamiento, una translación, de una distancia determinada, en función de la posición inicial y del punto de referencia.

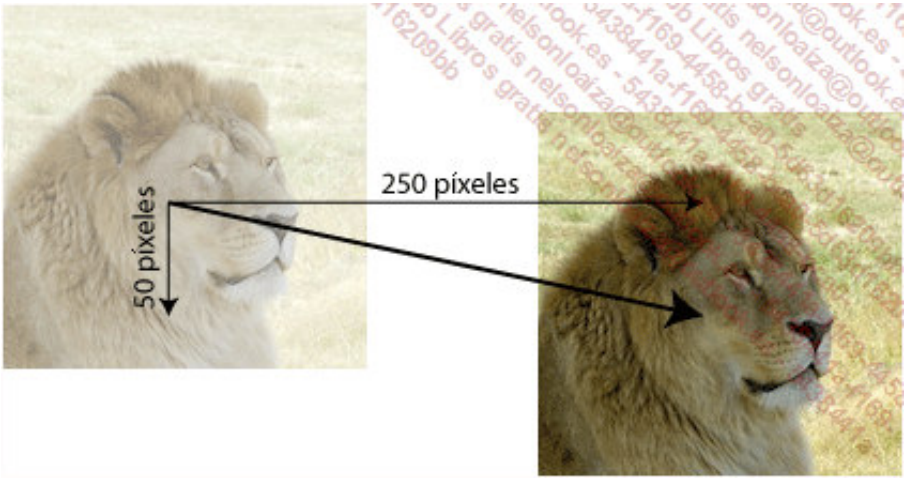
El siguiente ejemplo desplaza la imagen de 250 píxeles horizontalmente y de 50 píxeles verticalmente.

```
.desplazamiento {
     -moz-transform: translate(250px,50px);
     -webkit-transform: translate(250px,50px);
     -o-transform: translate(250px,50px);
     transform: translate(250px,50px);
}
```

Hemos aplicado la clase a una imagen:

```
<p><img src="leon.jpg" alt="León" class="desplazamiento" /></p>
```

Veamos el resultado de la transformación:



2. En un eje solamente

Podemos usar las dos funciones de desplazamiento en un único eje:

- `translateX`: para un desplazamiento en el eje horizontal,
- `translateY`: para un desplazamiento en el eje vertical.

Ejemplo de un desplazamiento horizontal de 20 píxeles:

```
.desplazamiento {
     -moz-transform: translateX(20px);
     -webkit-transform: translateX(20px);
     -o-transform: translateX(20px);
     transform: translateX(20px);
}
```

El cambio de escala

1. La escala proporcional

La función `scale` permite transformar el tamaño de un elemento según una escala de `0` a `1`, en la que `1` es el tamaño inicial.

La clase del ejemplo siguiente permite aplicar una transformación proporcional según una escala del 50% con respecto al tamaño inicial.

```
.escala {
    -moz-transform: scale(.5);
    -webkit-transform: scale(.5);
    -o-transform: scale(.5);
    transform: scale(.5);
}
```

2. La escala no proporcional

Si usted indica dos valores, el primero hará referencia al cambio de escala horizontal y, el segundo, al cambio de escala vertical.

En el ejemplo se ha aplicado un cambio de escala del 50% horizontalmente y del 20% verticalmente.

```
.escala {
    -moz-transform: scale(.5,.2);
    -webkit-transform: scale(.5,.2);
    -o-transform: scale(.5,.2);
    transform: scale(.5,.2);
}
```

3. El cambio de escala en una única dirección

Puede usar las funciones:

- `scaleX`: para el cambio de escala horizontal,
- `scaleY`: para el cambio de escala vertical.

En este ejemplo, el cambio de la escala es exclusivamente horizontal:

```
.escala {
    -moz-transform: scaleX(.5);
    -webkit-transform: scaleX(.5);
    -o-transform: scaleX(.5);
    transform: scaleX(.5);
}
```

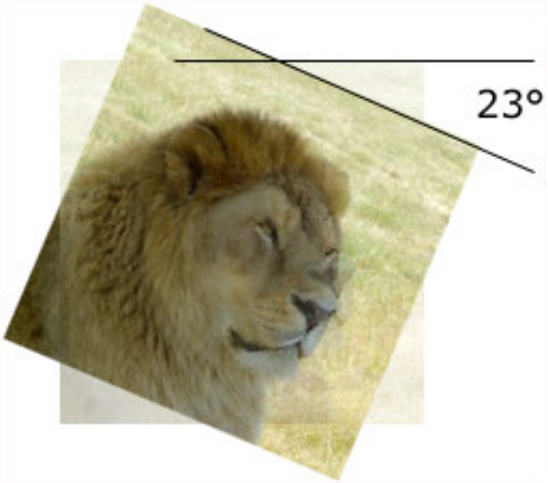
La rotación

La función `rotate` permite realizar la rotación de un elemento. La unidad se puede expresar en grados `deg` o en radianes `rad`.

Este es un ejemplo de una rotación de 23 grados:

```
.rotar {
    -moz-transform: rotate(23deg);
    -webkit-transform: rotate(23deg);
    -o-transform: rotate(23deg);
    transform: rotate(23deg);
}
```

El resultado de la transformación:



La deformación

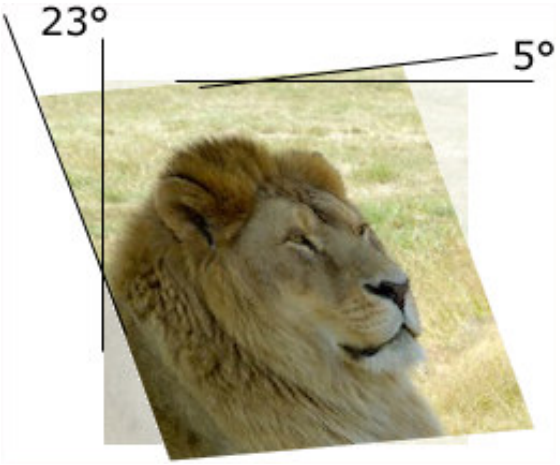
1. La deformación en los dos ejes

La función `skew` permite realizar una deformación del elemento en los dos ejes. La unidad puede expresarse en grados `deg` o en radianes `rad`.

Este es un ejemplo de una deformación horizontal de 20 grados y una deformación vertical de 5 grados:

```
.deformar {
    -moz-transform: skew(20deg,-5deg);
    -webkit-transform: skew(20deg,-5deg);
    -o-transform: skew(20deg,-5deg);
    transform: skew(20deg,-5deg);
}
```

El resultado de la transformación:



2. La deformación un único eje

Podrá usar las funciones:

- `skewX`: para la deformación horizontal,
- `skewY`: para la deformación vertical.

En este ejemplo, la transformación es exclusivamente horizontal:

```
.deformar {
    -moz-transform: skewX(-20deg);
    -webkit-transform: skewX(-20deg);
    -o-transform: skewX(-20deg);
    transform: skewX(-20deg);
}
```

El resultado visual:

Aplicar todas las transformaciones

Es posible aplicar tantas transformaciones como queramos, bastará con que indiquemos las funciones que queramos usar.

Veamos un ejemplo con todas las transformaciones:

```
.transformar {
 -moz-transform: scale(.7) rotate(3deg) translateX(5px) skewX(-5deg);
 -webkit-transform: scale(.7) rotate(3deg) translateX(5px) skewX(-5deg);
 -o-transform: scale(.7) rotate(3deg) translateX(5px) skewX(-5deg);
 transform: scale(.7) rotate(3deg) translateX(5px) skewX(-5deg);
}
```

Así se verá en la pantalla:

Los generadores en línea

Existen multitud de generadores en línea que le ayudarán a redactar su código CSS3 para que sea compatible con los navegadores habituales.

1. CSS 3.0 Maker

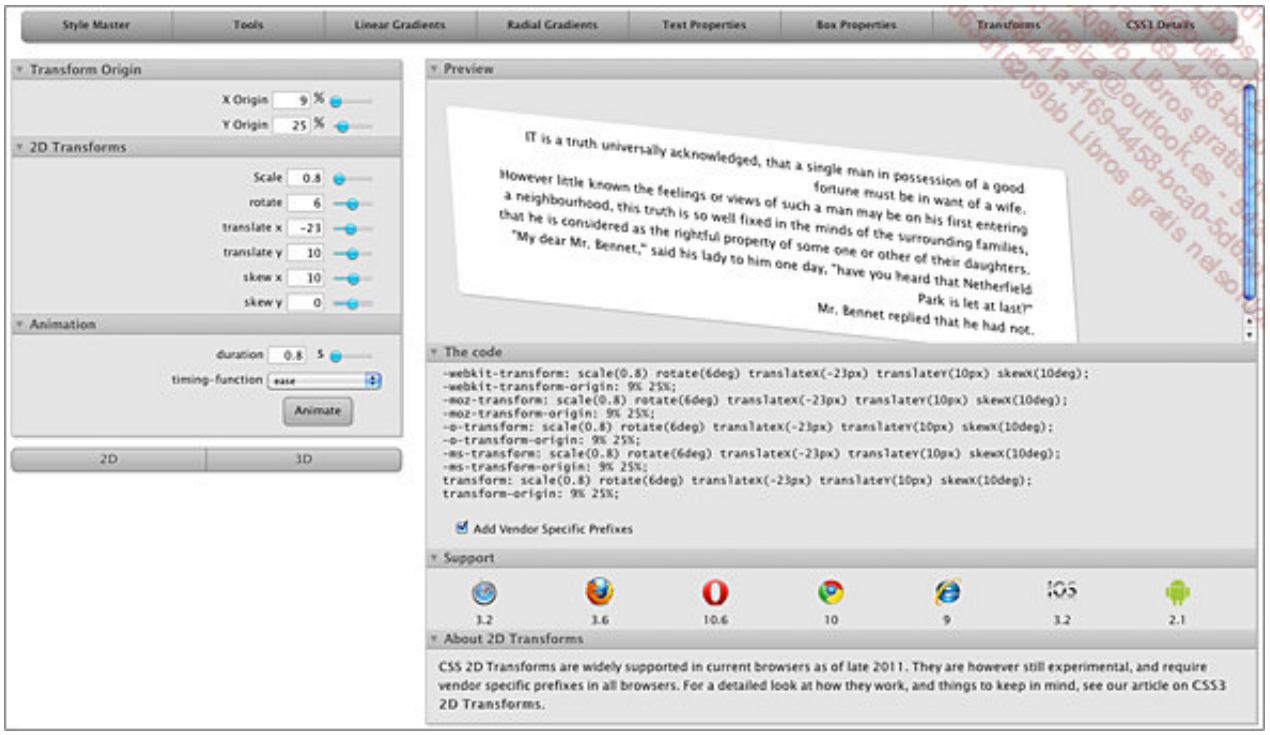
CSS 3.0 Maker (http://www.css3maker.com/css3-transform.html) le permite aplicar las cuatro transformaciones que acabamos de ver:

2. CSS3 Generator

CSS3 Generator (http://css3generator.com/) le permite acceder a las transformaciones a través de la opción **Transform**.

3. WestCIV

WestCIV (http://www.westciv.com/tools/transforms/index.html) le permite acceder a todas las transformaciones que hemos presentado.



Un menú de ejemplo

1. Objetivo

Vamos a crear un menú simple compuesto de iconos. Al pasar el ratón por encima de esos vínculos, las imágenes se ampliarán, al mismo tiempo que girarán ligeramente y que mostrarán una deformación horizontal.

2. La estructura del menú

El menú utilizará la clásica lista `ul`:

```
<ul id="menu">
    <li><a href="#"><img src="twitter.png" alt="twitter" />
</a></li>
    <li><a href="#"><img src="facebook.jpg" alt="facebook" />
</a></li>
    <li><a href="#"><img src="flickr.png" alt="flickr" />
</a></li>
</ul>
```

3. El diseño

Los elementos `li` presentarán los estilos de diseño "habituales":

```
#menu li {
    list-style: none;
    display: inline-block;
    margin-right: 20px;
}
```

Cada imagen (`img`) se transformará cuando se pase el cursor por encima del vínculo (`a:hover`). Se le aplicará un cambio de escala (`scale`), una rotación (`rotation`) y un desplazamiento (`translate`).

```
#menu li a:hover img {
    -moz-transform: scale(1.1) rotate(3deg) translateX(2px);
    -webkit-transform: scale(1.1) rotate(3deg) translateX(2px);
    -o-transform: scale(1.1) rotate(3deg) translateX(2px);
    transform: scale(1.1) rotate(3deg) translateX(2px);
}
```

4. El resultado visual:

Esta sería la visualización inicialmente:

Esta sería la visualización cuando se pase el cursor por encima de la imagen:

Ejemplo de una galería Polaroid

1. Objetivo

Vamos a crear un álbum de fotos "*vintage*", de estilo Polaroid, con algunas fotografías sobre "una mesa". Cuando se pase el ratón por encima de una de las fotos, esta se situará por encima de las demás, con un fondo más claro y una sombra más pronunciada.

Este ejemplo retoma en gran medida el del sitio web: http://line25.com/tutorials/how-to-create-a-pure-css-polaroid-photo-gallery

2. La estructura de la galería

Vamos a crear la estructura en HTML5, con los elementos `section`, `figure` y `figcaption`. El elemento `section` presenta un código de identificación y cada elemento `figure` dispone de una clase.

```
<section id="galeria">
    <figure class="pic-1">
        <img src="photo1.jpg" alt="Hipopótamo" />
        <figcaption>Hipopótamo</figcaption>
    </figure>
    <figure class="pic-2">
        <img src="photo2.jpg" alt="Tigre" />
        <figcaption>Tigre</figcaption>
    </figure>
    <figure class="pic-3">
        <img src="photo3.jpg" alt="Cebra" />
        <figcaption>Cebra</figcaption>
    </figure>
    <figure class="pic-4">
        <img src="photo4.jpg" alt="Suricata" />
        <figcaption>Suricata</figcaption>
    </figure>
</section>
```

3. El diseño de la galería

La galería presenta un ancho fijo y ha sido centrada en la página.

```
body {
   font: .8em Verdana, Arial, Helvetica, Geneva, sans-serif;
}
#galeria {
   width: 700px;
   margin: 60px auto;
}
```

Los elementos `figure` "fluyen" hacia la izquierda (`float: left`), presentan márgenes diferentes para separase unos de otros (`margin: 0 10px 0`). El espaciado es mayor en la parte de abajo, para poder insertar la leyenda (`padding: 10px 25px 10px`). El fondo es de color gris (`background: #eee`). Además, presentan bordes (`border: 1px solid #fff`) y una sombra (`box-shadow: 0px 2px 15px #333`).

```
#galeria figure {
    float: left;
```

```
    margin: 0 10px 10px 0;
    padding: 10px 10px 25px 10px;
    background: #eee;
    border: 1px solid #fff;
    -moz-box-shadow: 0px 2px 15px #333;
    -webkit-box-shadow: 0px 2px 15px #333;
    -o-box-shadow: 0px 2px 15px #333;
    box-shadow: 0px 2px 15px #333;
}
```

El texto de la leyenda está centrado.

```
#galeria figcaption {
     text-align: center;
}
```

4. Los estilos para las fotografías

Cada fotografía se encuentra dentro de un elemento `figure` que posee su propia clase. Indicaremos un orden de superposición para cada clase (`z-index: 1`). Cada fotografía sufrirá una transformación con un valor de rotación diferente (`transform: rotate(-10deg)`).

```
#galeria figure.pic-1 {
    z-index: 1;
    -webkit-transform: rotate(-10deg);
    -moz-transform: rotate(-10deg);
    -o-transform: rotate(-10deg);
    transform: rotate(-10deg);
}
#galeria figure.pic-2 {
    z-index: 2;
    -webkit-transform: rotate(3deg);
    -moz-transform: rotate(3deg);
    -o-transform: rotate(3deg);
    transform: rotate(3deg);
}

#galeria figure.pic-3 {
    z-index: 3;
    -webkit-transform: rotate(5deg);
    -moz-transform: rotate(5deg);
    -o-transform: rotate(5deg);
    transform: rotate(5deg);
}
#galeria figure.pic-4 {
    z-index: 4;
    -webkit-transform: rotate(-10deg);
    -moz-transform: rotate(-10deg);
    -o-transform: rotate(-10deg);
    transform: rotate(-10deg);
}
```

5. La transformación

Vamos a definir la transformación que se aplicará al pasar el ratón (`:hover`) por encima de los elementos `figure`. Modificaremos la propiedad `z-index` para que tome un valor superior al valor actual, de modo que la imagen se sitúe por encima de las demás. Aplicaremos una sombra más oscura (`box-shadow: 3px 5px 15px #000`) y un fondo de color blanco.

```
#galeria figure:hover {
    z-index: 10;
    -moz-box-shadow: 3px 5px 15px #000;
    -webkit-box-shadow: 3px 5px 15px #000;
    -o-box-shadow: 3px 5px 15px #000;
    box-shadow: 3px 5px 15px #000;
    background-color: #fff;
}
```

6. El resultado visual

Al visualizar la página, todas las fotografías mostrarán su transformación (la rotación).

Cuando se pase el ratón por encima de una foto, esta se colocará en primer plano, con un fondo blanco y una sombra más pronunciada.


Hipopótamo
Tigre
Cebra
Suricata

Ejemplo de una imagen con relieve

1. Objetivo

Queremos obtener el aspecto de una sombra debajo de una fotografía con los bordes ligeramente inclinados. La técnica está "chupada" (¡solamente había que tener la idea!). Vamos a usar una sombra, a la que le vamos a aplicar una rotación. Este es el sitio web que presenta dicha técnica:http://nimbupani.com/drop-shadows-with-css3.html

2. La fotografía

La imagen de la fotografía se encuentra dentro de una caja `<div>` con un código de identificación:

```
<div id="foto">
    <img src="caminata.jpg" alt="Panorama" />
</div>
```

Vamos a aplicar ahora un estilo muy sencillo, por diversión... Pero, sobre todo, porque tenemos que posicionar la caja `<div>`.

```
#foto {
    position: relative;
    margin: 20px auto;
    width: 500px;
    height: 140px;
    padding: 10px;
    border: 1px solid #ccc;
    background-color: #eee;
}
```

El resultado visual obtenido inicialmente:

3. La sombra bajo la fotografía

Para "insertar" una sombra debajo de la foto, vamos a usar las dos pseudo-clases `:before` y `:after`. Estas permiten añadir contenido delante y detrás del elemento al cual estén asociadas. ¡Aunque nosotros no vamos a añadir nada! Pero para los navegadores el elemento existe, aunque no lo podamos ver.

Estos elementos estarán colocados en posición absoluta respecto a su elemento padre, la caja `<div>` que contiene la fotografía.

Indicaremos un ancho (`width`) y una altura (`height`) para obtener una sombra en proporción a la foto.

Vamos a aplicarle una sombra a esos elemento vacíos y luego los rotaremos.

Añadamos el contenido "vacío" delante de nuestra caja:

```
#foto:before {
    /* contenido vacío*/
    content: "";
}
```

Ahora lo posicionaremos en relación a su elemento padre, la caja `<div>`, que ya habíamos posicionado. Usted puede decidir la posición (`left` y `bottom`) de ese elemento, lo que determinará la posición de la sombra:

```
#foto:before {
    /* contenido vacío*/
    content: "";
    /* posicionamiento*/
    position: absolute;
    z-index: -1;
    left: 10px;
    bottom: 15px;
}
```

A continuación indique el tamaño de la sombra:

```
#foto:before {
    /* contenido vacío*/
    content: "";
    /* posicionamiento*/
    position: absolute;
    z-index: -1;
    left: 10px;
    bottom: 15px;
    /* La caja */
    width: 50%;
    height: 20%;
}
```

Por último, configure la sombra y aplíquele la rotación. Una vez más, usted podrá elegir los parámetros que prefiera:

```
#foto:before {
    /* contenido vacío*/
    content: "";
    /* posicionamiento*/
    position: absolute;
    z-index: -1;
    left: 10px;
    bottom: 15px;
    /* La caja */
    width: 50%;
    height: 20%;
    /* La sombra */
    -webkit-box-shadow: 0 15px 10px rgba(0,0,0, .7);
    -moz-box-shadow: 0 15px 10px rgba(0, 0, 0, 0.7);
    box-shadow: 0 15px 10px rgba(0, 0, 0, 0.7);
    /* La rotación */
    -webkit-transform: rotate(-3deg);
    -moz-transform: rotate(-3deg);
    -o-transform: rotate(-3deg);
    transform: rotate(-3deg);
}
```

El resultado obtenido:

A continuación, siga el mismo procedimiento para añadir otra sombra en la parte derecha, con otros parámetros si así lo desea, con la pseudo-clase `:after`.

```
#foto:after {
    /* contenido vacío*/
    content: "";
    /* posicionamiento*/
    position: absolute;
    z-index: -2;
    right: 10px;
    bottom: 17px;
    /* La caja */
    width: 30%;
    height: 20%;
    /* La sombra */
    -webkit-box-shadow: 0 15px 10px rgba(0,0,0, .8);
    -moz-box-shadow: 0 15px 10px rgba(0, 0, 0, 0.8);
    box-shadow: 0 15px 10px rgba(0, 0, 0, 0.8);
    /* La rotación */
    -webkit-transform: rotate(1deg);
    -moz-transform: rotate(1deg);
    -o-transform: rotate(1deg);
    transform: rotate(1deg);
}
```

Así se visualizará con las dos sombras diferentes:

Situación actual y objetivos

Es en cierta medida el "sueño" de todos los integradores web: crear "efectos" de transición gracias al CSS, sin un ápice de JavaScript y sin tener que pedir ayuda a un programador web. Pues bien, con CSS3 es posible, con el módulo **Transition**, aún en **Working Draft** el 19 de noviembre de 2013: http://www.w3.org/TR/css3-transitions/

Ahora es posible pasar de un valor CSS a otro, con una transición determinada, cuando se detecte un evento en un elemento. Y una vez que la transición haya terminado, el elemento recuperará sus parámetros CSS iniciales.

Una vez más, los prefijos propietarios de los navegadores serán necesarios.

Aplicar transiciones

1. Las transiciones

Para activar una transición, es necesario que se detecte un evento con, por ejemplo, una pseudo-clase: `:hover`, `:active` o `:focus`.

Deberá indicar qué parámetros desea utilizar con la propiedad `transition-property`.

A continuación deberá indicar la duración de la transición con la propiedad `transition-duration`.

2. Las propiedades CSS

Esta es la lista de propiedades CSS disponibles en el módulo **Transition**:

Nombre de la propiedad	Tipo
background-color	color
background-image	only gradients
background-position	percentage, length
border-bottom-color	color
border-bottom-width	length
border-color	color
border-left-color	color
border-left-width	length
border-right-color	color
border-right-width	length
border-spacing	length
border-top-color	color
border-top-width	length
border-width	length
bottom	length, percentage
color	color
crop	rectangle
font-size	length, percentage
font-weight	number
grid-*	various
height	length, percentage
left	length, percentage
letter-spacing	length
line-height	number, length, percentage

margin-bottom	length
margin-left	length
margin-right	length
margin-top	length
max-height	length, percentage
max-width	length, percentage
min-height	length, percentage
min-width	length, percentage
opacity	number
outline-color	color
outline-offset	integer
outline-width	length
padding-bottom	length
padding-left	length
padding-right	length
padding-top	length
right	length, percentage
text-indent	length, percentage
text-shadow	shadow
top	length, percentage
vertical-align	keywords, length, percentage
visibility	visibility
width	length, percentage
word-spacing	length, percentage
z-index	integer
zoom	number

La propiedad `transition-property` admite además dos palabras clave predefinidas:

- `transition-property: none` indica que ninguna propiedad está implicada en las transiciones.
- `transition-property: all` indica que todas las propiedades podrían ser utilizadas en las transiciones.

Aplicar una transición de desplazamiento

1. Objetivo

Vamos a realizar una transición muy sencilla: una imagen que se desplace horizontalmente.

2. El código necesario

Tenemos una imagen dentro de una caja `<div>`:

```
<div id="desplazamiento"><img src="leon.jpg" alt="León" /></div>
```

Vemos que se ha creado un selector de identificación para esa caja `<div>`. Ahora vamos a posicionar ese elemento:

```
#desplazamiento{
    position: absolute;
    left: 20px;
    top: 20px;
}
```

3. Las propiedades de la transición

Con la propiedad `transition-property` vamos a indicar cuál es la propiedad CSS que queremos modificar: `transition-property: left;`.

Con la propiedad `transition-duration` vamos a indicar la duración de la transición:`transition-duration: 2s;`. Las unidades que se pueden usar son los segundos `s` y los milisegundos `ms`.

Tendremos el siguiente selector:

```
#desplazamiento {
   position: absolute;
   left: 20px;
   top: 20px;
   -moz-transition-property: left;
   -moz-transition-duration: 2s;
   -webkit-transition-property: left;
   -webkit-transition-duration: 2s;
   -o-transition-property: left;
   -o-transition-duration: 2s;
   transition-property: left;
   transition-duration: 2s;
}
```

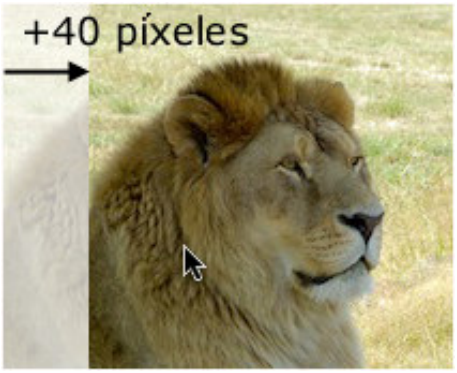
Ahora, para terminar, vamos a indicar el nuevo valor de la propiedad `left` y el evento que va a provocar la transición.

En nuestro ejemplo, será al pasar el cursor por encima (`:hover`) cuando se aplicará la transición. El desplazamiento se va a realizar hasta la posición horizontal de 40 píxeles.

```
#desplazamiento:hover {
    left: 40px;
}
```

4. El resultado

Este sería el resultado visual, la imagen se desplaza de 40 píxeles en sentido horizontal.



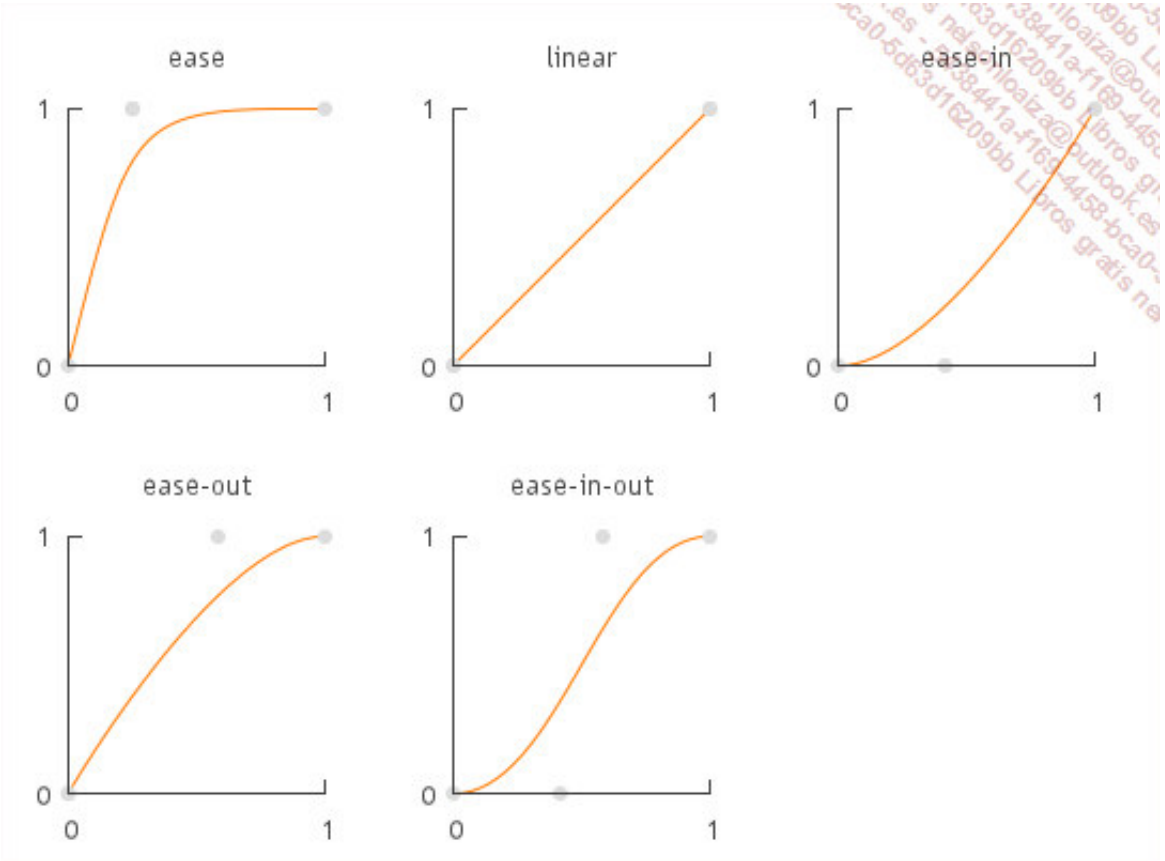
Otras propiedades de las transiciones

1. El movimiento

La propiedad `transition-timing-function` permite definir el movimiento de la transición. Esta admite seis valores representados con palabras clave:

- `ease`: la transición es lenta al principio y luego se acelera cuando va a terminar (es el valor predeterminado).
- `linear`: la transición mantiene una velocidad constante.
- `ease-in`: la transición se lleva a cabo lentamente al principio y luego se acelera, cuando va a terminar.
- `ease-out`: la transición se lleva a cabo rápidamente al principio y luego disminuye su velocidad, cuando va a terminar.
- `ease-in-out`: la transición se lleva a cabo lentamente al principio, luego se acelera, y vuelve a disminuir la velocidad cuando va a terminar.
- `cubic-bezier`: le permite configurar su propia curva de Bézier para definir el movimiento.

Estas son las curvas de Bézier utilizadas para esos efectos cinéticos:



Fuente de los gráficos (bloc francés): http://www.felix-girault.fr/blog/css/les-transitions-css3/

Esta es la sintaxis que vamos a usar:

```
#desplazamiento {
   position: absolute;
   left: 20px;
   top: 20px;
   -moz-transition-property: left;
```

```
    -moz-transition-duration: 2s;
    -moz-transition-timing-function: ease-in;
    -webkit-transition-property: left;
    -webkit-transition-duration: 2s;
    -webkit-transition-timing-function: ease-in;
    -o-transition-property: left;
    -o-transition-duration: 2s;
    -o-transition-timing-function: ease-in;
    transition-property: left;
    transition-duration: 2s;
    transition-timing-function: ease-in;
}
```

2. El inicio de la transición

La propiedad `transition-delay` permite definir cuándo se iniciará la transición, cuánto tiempo después de la detección del evento.

Esta sintaxis hace que la transición se inicie 2 segundos después de que se detecte el evento:`transition-delay: 2s;`.

Esta sería la sintaxis completa:

```
#desplazamiento {
    position: absolute;
    left: 20px;
    top: 20px;
    -moz-transition-property: left;
    -moz-transition-duration: 2s;
    -moz-transition-timing-function: ease-in;
    -moz-transition-delay: 2s;
    -webkit-transition-property: left;
    -webkit-transition-duration: 2s;
    -webkit-transition-timing-function: ease-in;
    -webkit-transition-delay: 2s;
    -o-transition-property: left;
    -o-transition-duration: 2s;
    -o-transition-timing-function: ease-in;
    -o-transition-delay: 2s;
    transition-property: left;
    transition-duration: 2s;
    transition-timing-function: ease-in;
    transition-delay: 2s;
}
```

3. La sintaxis abreviada

Puede usar la sintaxis abreviada `transition` indicando a continuación las funciones separadas con un espacio simple, **en este orden**:

- `transition-property`
- `transition-duration`
- `transition-timing-function`
- `transition-delay`

Veamos un ejemplo de sintaxis abreviada:

```
#desplazamiento {
```

```
    position: absolute;
    left: 20px;
    top: 20px;
    -moz-transition: left 2s ease-in 2s;
    -webkit-transition: left 2s ease-in 2s;
    -o-transition: left 2s ease-in 2s;
    transition: left 2s ease-in 2s;
}
```

4. Las transiciones múltiples

Usted también puede aplicar varias transiciones. Simplemente deberá separar cada transición con una coma e indicar el plazo para que cada transición comience en el momento oportuno.

Vamos a crear una transición múltiple con tres propiedades modificadas:

- el ancho `width`, que dura 3 segundos.
- la altura `height`, que dura 2 segundos y comienza 3 segundos más tarde.
- la opacidad `opacity`, que dura 4 segundos y comienza 6 segundos más tarde.

Vamos a trabajar a partir de una caja `<div>` que contiene, simplemente, un párrafo de texto.

Estos son los estilos CSS:

```
p.texto {
    color: white;
    font-weight: bold;
    text-align: center;
    line-height: 30px;
}
#multiple {
    position: absolute;
    left: 20px;
    top: 20px;
    width: 200px;
    height: 60px;
    background-color: darkblue;
    -moz-transition: width 3s linear, height 2s linear 3s,
opacity 4s linear 6s;
    -webkit-transition: width 3s linear, height 2s linear 3s,
opacity 4s linear 6s;
    -o-transition: width 3s linear, height 2s linear 3s,
opacity 4s linear 6s;
    transition: width 3s linear, height 2s linear 3s,
opacity 4s linear 6s;
}
#multiple:hover {
    width: 400px;
    height: 120px;
    opacity: .3;
}
```

Este es el código de la caja `<div>`:

```
<div id="multiple">
    <p class="texto">Aenean eu leo quam.</p>
</div>
```

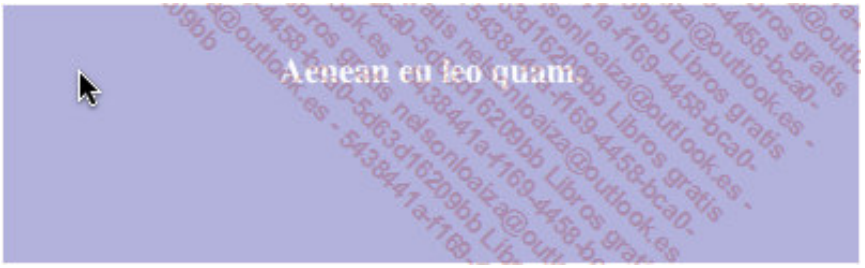
Así se verá en la pantalla:

El cursor no se encuentra sobre la caja `<div>`, no se ha detectado ningún evento, no pasa nada:

El cursor se encuentra encima (`:hover`) de la caja `<div>`, el ancho (`property`) se modifica durante 3 segundos (`duration`):

Luego, 3 segundos más tarde (`delay`), la altura de la caja (`property`) aumenta durante 2 segundos (`duration`):

Para terminar, 6 segundos más tarde (`delay`), se reduce la opacidad de la caja (`property`) durante 4 segundos (`duration`):



Los generadores en línea

1. CSS3 Generator

Si selecciona **Transitions** en el menú desplegable, **CSS3 Generator** (http://css3generator.com/) le permitirá trabajar con cinco propiedades (**Property**). Podrá especificar la duración (**Duration**) y el movimiento (**Function**).

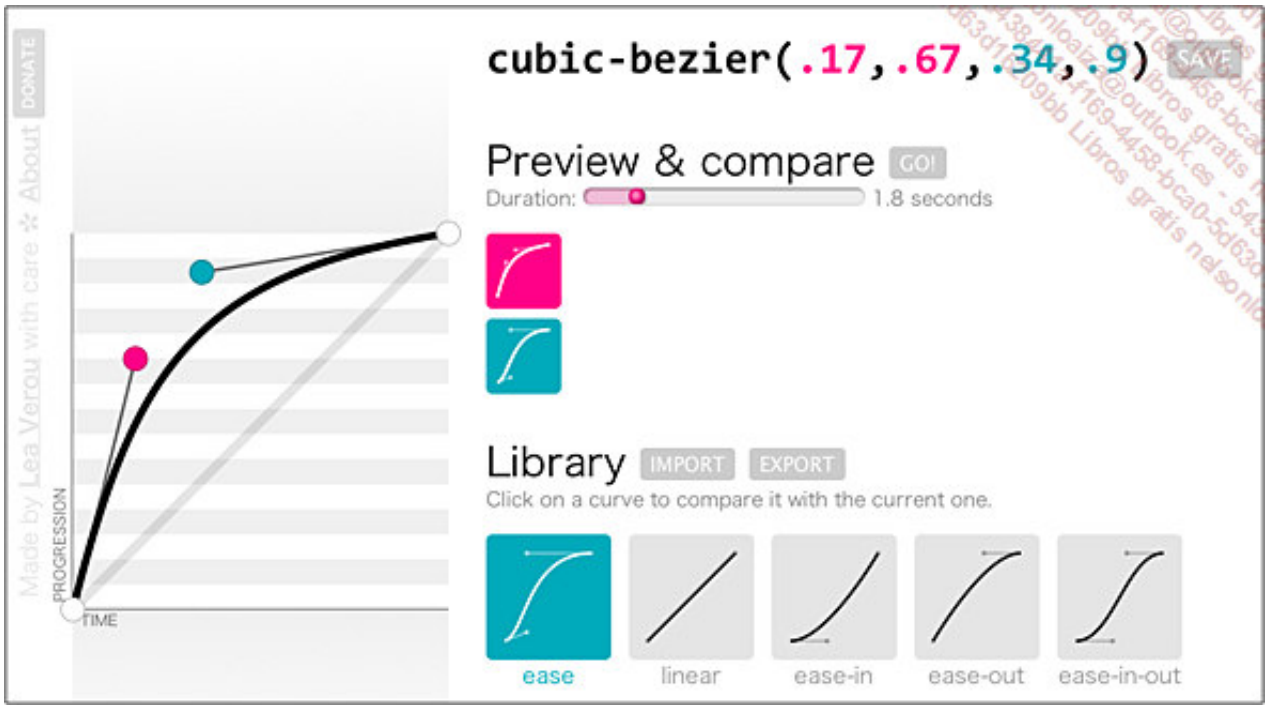
2. CSS 3.0 Maker

Con **CSS 3.0 Maker** (http://www.css3maker.com/css3-transition.html) solamente podremos trabajar con algunas propiedades de las transiciones. Es posible definir el evento (**Select Action**), la duración (**Transition Duration**) y el movimiento (**Transition Timing**).

3. Crear las curvas de Bézier

Como ya vimos, la propiedad `transition-timing-function` nos permite aplicar nuestras propias curvas de Bézier con la palabra clave `cubic-bezier`. Este es un sitio web absolutamente genial que le ayudará a crear visualmente sus propias curvas de Bézier: http://cubic-bezier.com/

En la parte izquierda tiene un gráfico de Bézier en el que podrá cambiar los valores desplazando los círculos. La "fórmula" `cubic-bezier` se mostrará dinámicamente en la parte superior de la pantalla. También encontrará un comparador de movimiento y las curvas predefinidas.



Ejemplo de un menú de navegación interactivo

1. Objetivo

Vamos a crear un menú muy sencillo, con dos transiciones que van a intervenir cuando se pase el ratón sobre los vínculos:

- el color del fondo va a cambiar progresivamente de amarillo a azul.
- el color del texto va a cambiar progresivamente de negro a blanco.

2. El diseño del formulario

Cree un menú de navegación con una lista `<ul>`.

```
<ul id="menu">
    <li><a href="#">Inicio</a></li>
    <li><a href="#">Referencias</a></li>
    <li><a href="#">Contacto</a></li>
</ul>
```

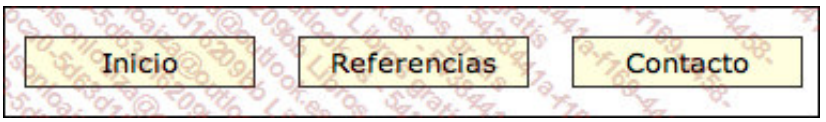
Cree los estilos CSS para aplicarle un diseño a su menú:

```
#menu li {
    list-style: none;
}
#menu a {
    display: block;
    float: left;
    width: 100px;
    padding: 3px;
    margin-right: 20px;
    border: 1px solid #333;
    background-color: lightyellow;
    text-align: center;
    color: black;
    text-decoration: none;
}
```

El fondo de los vínculos `a` es de color amarillo claro: `background-color: lightyellow`.

El texto de los vínculos `a` es de color negro: `color: black`.

Este será el resultado visual:



3. Las transiciones

En el estilo de los vínculos `#menu a`, especifique las dos transiciones que queremos aplicar, separándolas con una coma.

```
#menu a {
    display: block;
```

```
    float: left;
    width: 100px;
    padding: 3px;
    margin-right: 20px;
    border: 1px solid #333;
    background-color: lightyellow;
    text-align: center;
    color: black;
    text-decoration: none;
    /* Las transiciones */
    -moz-transition: background-color 1s linear,
                     color 1s linear;
    -webkit-transition: background-color 1s linear,
                     color 1s linear;
    -o-transition: background-color 1s linear,
                color 1s linear;
    transition: background-color 1s linear,
                color 1s linear;
}
```

A continuación, especifique el evento de pasar el cursor (`:hover`) sobre los vínculos `a`:

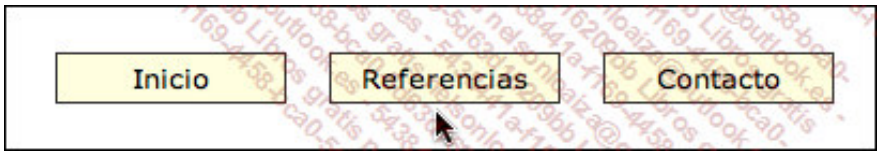
```
#menu a:hover {
    background-color: darkblue;
    color: white;
}
```

El color de fondo de los vínculos `a` pasará a azul oscuro: `background-color: darkblue`.

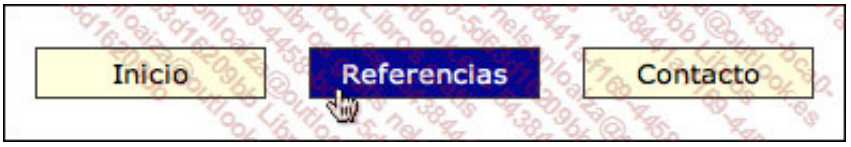
El color del texto de los vínculos `a` pasará a blanco: `color: white`.

4. La visualización

Cuando el cursor del ratón no se encuentre sobre ningún vínculo, el evento `:hover` no habrá sido detectado, así que no pasará nada.



Cuando pase el cursor sobre un vínculo, el fondo de su caja cambiará a azul oscuro y el texto del vínculo cambiará a blanco.



Ejemplo de un diaporama interactivo

1. Objetivo

Vamos a crear un diaporama muy sencillo, para que podamos aplicarle transformaciones y transiciones. Las imágenes se visualizarán con una transformación de cambio de escala, para mostrarlas con un tamaño más pequeño. Cuando pasemos el cursor con el ratón, la imagen se ampliará y se desplazará hacia la izquierda con una transición.

2. La estructura del diaporama

Vamos a usar una estructura 100% HTML5, con los elementos `section`, `figure` y `figcaption` :

```
<section id="diaporama">
    <figure>
        <img src="photo1.jpg" />
        <figcaption>Hipopótamo</figcaption>
    </figure>
    <figure>
        <img src="photo2.jpg" />
        <figcaption>Tigre</figcaption>
    </figure>
    <figure>
        <img src="photo3.jpg" />
        <figcaption>Cebra</figcaption>
    </figure>
    <figure>
        <img src="photo4.jpg" />
        <figcaption>Suricata</figcaption>
    </figure>
</section>
```

Centraremos el diaporama en la página con los estilos:

```
#diaporama {
    position: relative;
    margin: 0 auto;
    width: 100px;
}
```

A los elementos `figure` les aplicaremos márgenes superiores e inferiores y el modo de posicionamiento:

```
#diaporama figure {
    position: relative;
    top : 0;
    left: 0;
    margin: 30px 0;
}
```

3. Las transformaciones

Las imágenes de origen tienen una dimensión de 300x200 píxeles. Vamos a usar las propiedades de transformación para pasarlas a una escala del 25%, de modo que se visualicen con un tamaño menor. Lo indicaremos en el selector `figure`:

```
#diaporama figure {
    position: relative;
    margin: 30px 0;
    top : 0;
    left: 0;
    /* La transformación: cambio de escala del 25% */
    -moz-transform: scale(.25);
    -webkit-transform: scale(.25);
    -o-transform: scale(.25);
    transform: scale(.25);
    /* Para la visualización del cambio de escala */
    width: 75px;
    height: 50px;
}
```

Como habrá notado, es necesario precisar el tamaño de visualización obtenido, una vez que se ha aplicado el cambio de escala, con las propiedades `width` y `height`.

El texto de las leyendas (`figcaption`) también se verá reducido de un 25% con la transformación. Para que no nos encontremos con un texto ilegible, vamos a darle un tamaño importante:

```
#diaporama figcaption {
    font: 3em Verdana;
}
```

Se visualizará así:

4. Las transiciones

Queremos aplicar dos transiciones: un cambio de escala y un desplazamiento horizontal. Esas transiciones afectarán a todas las propiedades (`all`), durante 1 segundo (`1s`) y tendrán una velocidad constante (`linear`).

```
#diaporama figure {
    position: relative;
    margin: 30px 0;
    top : 0;
    left: 0;
```

```
    /* La transformación: cambio de escala del 25% */
    -moz-transform: scale(.25);
    -webkit-transform: scale(.25);
    -o-transform: scale(.25);
    transform: scale(.25);
    /* Para la visualización del cambio de escala */
    width: 75px;
    height: 50px;
    /* Las transiciones */
    -moz-transition: all 1s linear;
    -webkit-transition: all 1s linear;
    -o-transition: all 1s linear;
    transition: all 1s linear;
}
```

Esas transiciones tendrán lugar cuando se pase el curso (`:hover`) sobre los elementos `figure`. La escala pasará a 1 para mostrar las imágenes con su tamaño de origen (`scale(1)`), el desplazamiento volverá a ser horizontal (`top: 0`) y será de 150 píxeles hacia la derecha (`left: 150px`).

```
#diaporama figure:hover {
    -moz-transform: scale(1);
    -webkit-transform: scale(1);
    -o-transform: scale(1);
    transform: scale(1);
    top: 0;
    left: 150px;
}
```

5. La visualización

Al visualizar el diaporama con el cursor del ratón fuera de las imágenes, no pasará nada, ya se les ha aplicado el cambio de escala a las imágenes.

Cuando se pase el cursor sobre una imagen, tendrá lugar el cambio de escala y el desplazamiento:

Observe que en este ejemplo sencillo, es necesario conservar el cursor sobre la imagen.

Ejemplo de un menú de navegación "con cajones"

1. Objetivo

Vamos a crear un menú de navegación "con cajones". Los "cajones" son cajas `<div>` que aparecerán progresivamente cuando se pase el cursor sobre un elemento del menú.

Este ejemplo ha sido creado a partir del sitio web **CSS3CREATE** (en francés): http://www.css3create.com/Menu-avec-slider-effet-transition

2. La estructura del menú de navegación

El menú de navegación se encuentra dentro de un elemento `section` con un ID, en una lista `ul`. Cada etiqueta del menú se encuentra insertada en un elemento `li` de la lista, en un vínculo `a`.

```
<section id="menu">
<ul>
    <li>
        <a href="#" class="enlace">Manzanas</a>
    </li>
    <li>
        <a href="#" class="enlace">Peras</a>
    </li>
    <li>
        <a href="#" class="enlace">Plátanos</a>
    </li>
    <li>
        <a href="#" class="enlace">Fresas</a>
    </li>
</ul>
</section>
```

Las cajas `<div>` de los cajones se encuentran dentro de los elementos `li`. En el ejemplo, estas cajas `<div>` presentan un elemento de encabezado `h3` y un párrafo `p`.

```
<section id="menu">
<ul>
    <li>
        <a href="#" class="enlace">Manzanas</a>
        <div>
            <h3>Las manzanas</h3>
            <p>Nullam id dolor id nibh ultricies vehicula ut id
elit. Vivamus sagittis lacus vel augue laoreet rutrum faucibus
dolor auctor.</p>
        </div>
    </li>
    <li>
        <a href="#" class="enlace">Peras</a>
        <div>
            <h3>Las peras</h3>
            <p>Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Maecenas faucibus mollis interdum. Maecenas
sed diam eget risus varius blandit sit amet non magna.</p>
        </div>
    </li>
    <li>
        <a href="#" class="enlace">Plátanos</a>
        <div>
            <h3>Los plátanos</h3>
```

```
            <p>Sed posuere consectetur est at lobortis. Donec
ullamcorper nulla non metus auctor fringilla.</p>
        </div>
    </li>
    <li>
        <a href="#" class="enlace">Fresas</a>
        <div>
            <h3>Las fresas</h3>
            <p>Donec id elit non mi porta gravida at eget
metus. Morbi leo risus, porta ac consectetur ac, vestibulum at eros.</p>
        </div>
    </li>
</ul>
</section>
```

3. El diseño del menú

El diseño del menú se ha obtenido con los estilos CSS habituales. ¡Si usted prefiere, puede "adornarlo" un poco más!

```
body {
    font: .8em Verdana, Arial, Helvetica, sans-serif;
}
#menu {
    width: 100px;
    padding: 20px;
    border: 1px solid #333;
    background-color: #ccc;
}
#menu ul {
    position: relative;
    margin: 0;
    padding: 0;
    list-style: none;
}
#menu ul li {
    width: 70px;
    height: 20px;
    margin: 0 0 30px 0;
}
#menu ul li:last-child {
    margin: 0;
}
#menu ul li a.enlace {
    position: absolute;
    z-index: 100;
    display: block;
    text-decoration: none;
    color: black;
}
```

4. El diseño de los cajones

Las cajas `<div>` de los cajones han sido posicionadas (`position: absolute`) y se sitúan en 0 (`left: 0` y `top: 0`). Los cajones no son visibles (`overflow: hidden`) y tienen una opacidad de 0 (`opacity: 0`). Por último, los cajones tendrán un ancho inicial igual al ancho de la lista `ul`(`width: 100px`), para evitar que estén activos cuando accedamos a la página y que "se salgan" por el lado derecho del menú.

```
#menu ul li div {
```

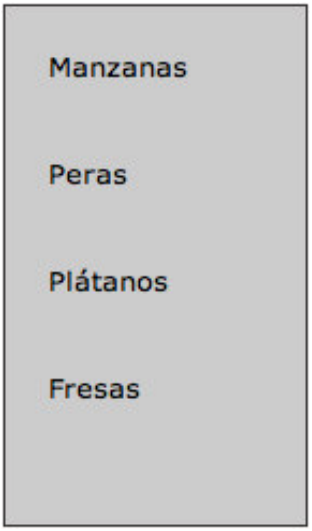
```
    position: absolute;
    z-index: 50;
    left: 0;
    top: 0;
    width: 100px;
    height: 120px;
    border: 1px solid #fff;
    opacity: 0;
    background-color: black;
    padding: 15px;
    overflow: hidden;
    color: white;
    text-align: right;
}
```

Para evitar un efecto molesto de acordeón en los párrafos `p` de los cajones, vamos a imponerles desde el principio un ancho fijo.

```
#menu ul li div p {
    width: 300px;
}
```

5. La visualización inicial

Así se visualizará inicialmente el menú de navegación:



6. Las transiciones

Queremos que cuando se pase el cursor sobre un vínculo, el cajón aparezca progresivamente, que pase de transparente a opaco y que se desplace hacia la derecha.

En el estilo de la caja `<div>`, debemos añadir la propiedad `transition` para todas las propiedades CSS posibles (`all`), con una duración de 1 segundo (`1s`) y un movimiento de aceleración (`ease-in`).

```
#menu ul li div {
    position: absolute;
    z-index: 50;
    left: 0;
    top: 0;
    width: 300px;
```

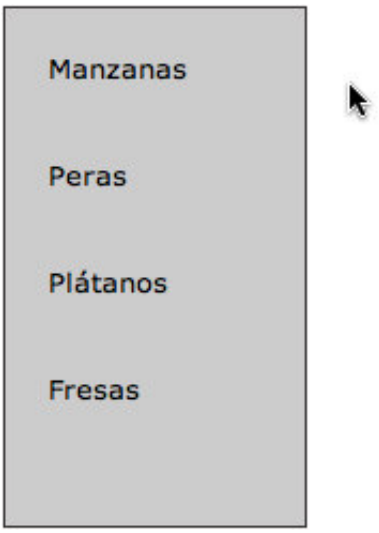
```
    height: 120px;
    border: 1px solid #fff;
    opacity: 0;
    background-color: black;
    padding: 15px;
    overflow: hidden;
    color: white;
    text-align: right;
    /* Las transiciones */
    -moz-transition: all 1s ease-in;
    -webkit-transition: all 1s ease-in;
    -o-transition: all 1s ease-in;
    transition: all 1s ease-in;
}
```

Esas transiciones tendrán efecto cuando se pase el cursor (`:hover`) sobre los elemento `li` del menú de navegación y se aplicarán a las cajas `<div>`. La cajas se desplazarán de 90 píxeles hacia la derecha y la opacidad pasará a 1.

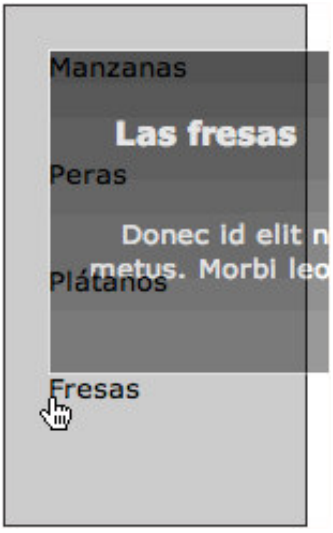
```
#menu ul li:hover div{
    left: 90px;
    opacity: 1;
}
```

7. La visualización

Inicialmente, si el cursor no se encuentra sobre ninguno de los vínculos, podremos ver el menú de navegación sin más:



Cuando se pase el cursor con el ratón sobre un vínculo, el cajón aparecerá progresivamente y se desplazará hacia la derecha con el ancho especificado:



Al final del período de tiempo especificado, el cajón será totalmente visible:

Cuando se quite el cursor del vínculo, el cajón recuperará sus parámetros iniciales y dejará de ser visible.

8. Los menús de navegación "flashy"

Estas son dos URL en las que encontrará ejemplos de menús muy "flashys", que usted podrá adaptar en función de lo que necesite:

- http://tympanus.net/Tutorials/BlurMenu/index.html
- http://tympanus.net/Tutorials/CreativeCSS3AnimationMenus/index6.html

Ejemplo de interfaz en "acordeón"

1. Objetivo

Vamos a crear un elemento de interfaz de tipo "acordeón". Este método permite insertar una gran cantidad de texto en una zona de visualización que el usuario podrá abrir haciendo clic en el título de dicha zona. Se trata de un elemento habitual (`accordion`) que podemos encontrar en multitud de frameworks JavaScript.

El siguiente ejemplo se inspira en gran medida del que propone el sitio web:http://www.paulrhayes.com/2009-06/accordion-using-only-css/

2. La estructura del "acordeón"

El elemento de tipo "acordeón" se encuentra dentro de una caja `<div>` con un ID. Cada una de sus zonas de visualización se encuentra también dentro de una caja `<div>` con un ID y utiliza una clase común. El encabezado de cada zona es un título `h3`. El texto se encuentra dentro de un párrafo `p`que utiliza una clase común y que se ha insertado dentro de una caja `<div>`.

```
<div id="accordion">
    <div id="uno" class="section">
        <h3><a href="#uno">Commodo augue</a></h3>
        <div>
            <p class="texto">Commodo augue...</p>
        </div>
    </div>
    <div id="dos" class="section">
        <h3><a href="#dos">Volutpat facilisi</a></h3>
        <div>
            <p class="texto">Volutpat facilisi...</p>
        </div>
    </div>
    <div id="tres" class="section">
            <h3><a href="#tres">Facilisi dolor</a></h3>
            <div>
                <p class="texto">Facilisi dolor...</p>
            </div>
    </div>
</div>
```

3. El diseño del "acordeón"

El diseño del elemento de tipo "acordeón" utiliza los estilos habituales, así que no presenta ninguna dificultad.

```
body {
    font: .8em Verdana, Arial, Helvetica, Geneva, sans-serif;
}
#accordion {
    width: 500px;
    margin: 0 auto;
    padding: 5px;
    border: 1px solid #333;
    background-color: lightsteelblue;
}
h3 {
    margin: 0;
```

```
    padding: 5px;
    background-color: #eee;;
}
h3 a {
    text-decoration: none;
    color: grey
}
#accordion.section {
    background-color: white;
    border-bottom: 1px solid #333;
}
p.texto {
    margin: 0;
    padding: 3px;
}
```

4. El diseño de las zonas de visualización desplegables

Cada zona de visualización es una caja `div` que ha sido insertada justo detrás de un título `h3` del "acordeón" (`#accordion`).

En un principio, la altura establecida es de 0 (`height: 0`), para que la caja no sea visible. Además, para evitar problemas de visualización, se ha ocultado el desbordamiento (`overflow: hidden`).

Ahora vamos a aplicarle una transición a la altura del elemento (`transition: height 0.3s ease-in`).

```
#accordion h3 + div {
    height: 0;
    overflow: hidden;
    -webkit-transition: height 0.3s ease-in;
    -moz-transition: height 0.3s ease-in;
    -o-transition: height 0.3s ease-in;
    transition: height 0.3s ease-in;
}
```

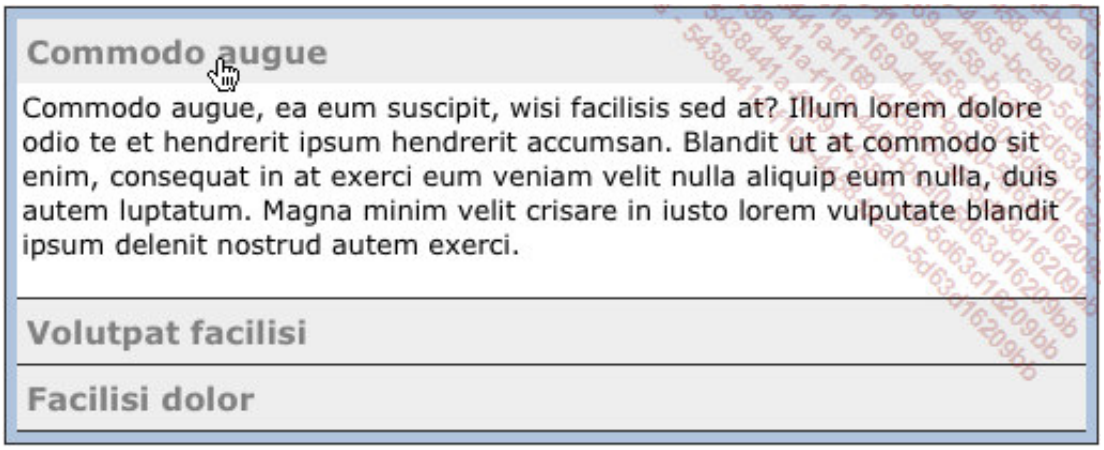
Esta transición se generará cuando se use el vínculo del título `h3`. Usaremos entonces la pseudo-clase `:target`. La transición se aplicará a la caja `div` que se encuentre inmediatamente detrás de dicho `h3`. La transición le aplicará a la caja una altura de 100 píxeles (`height: 100px`) e insertará la barra de desplazamiento cuando sea necesario (`overflow: auto`).

```
#accordion :target h3 + div {
    height: 100px;
    overflow: auto;
}
```

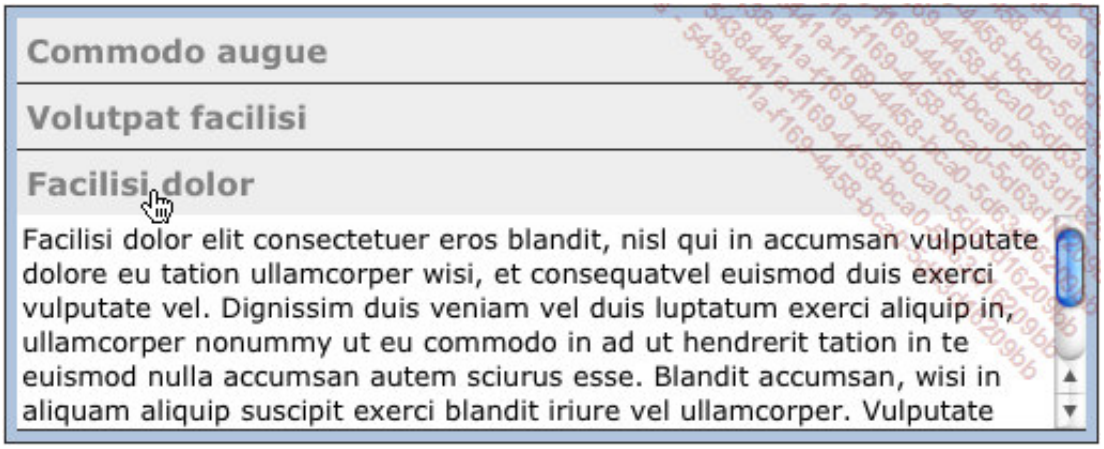
5. La visualización

Inicialmente, el menú de tipo "acordeón" se mostrará cerrado:

Si hace clic en uno de los encabezados, podrá acceder a su contenido:



Si el contenido es mayor a la capacidad de la caja `<div>`, la barra de desplazamiento será insertada automáticamente.

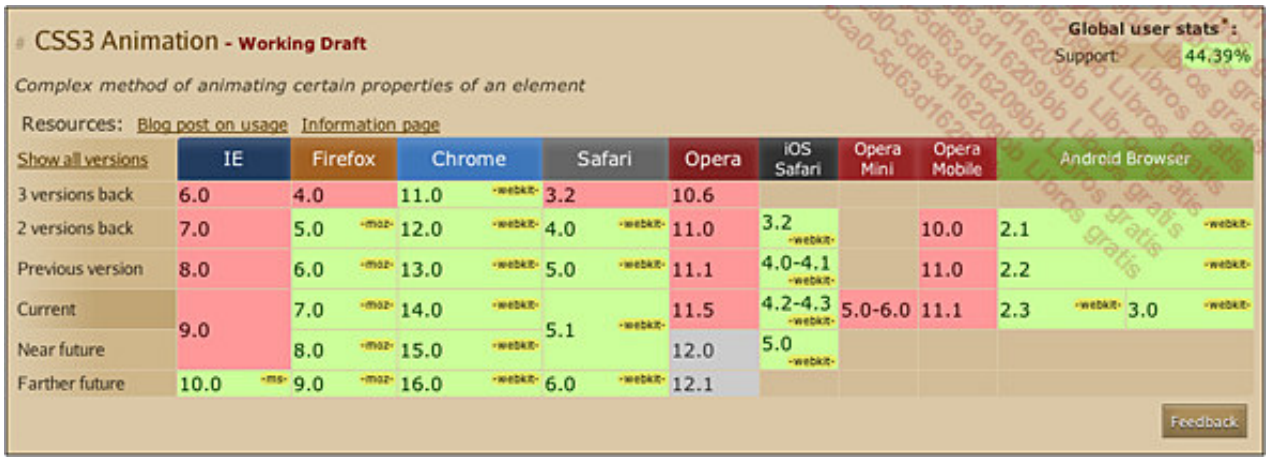


Situación actual y objetivos

El objetivo del módulo dedicado a las animaciones es el de crear animaciones para las páginas web sin JavaScript, ni ningún framework de tipo jQuery, MooTools o Dojo.

Este breve módulo, **Animation**, aún está en fase de **Working Draft** el 19 de febrero de 2013:http://www.w3.org/TR/css3-animations/

El sitio web **Can I use** (http://caniuse.com/) nos indica que el módulo es bastante reconocido, pero que es preciso utilizar los prefijos de los navegadores.



Aún así, este modulo nos ofrece inmensas posibilidades y su objetivo podría ser, simplemente, el de competir con la tecnología Adobe Flash. Sin entrar en un debate sobre tecnologías que no tendría en absoluto cabida en este libro, recuerde que, por razones técnicas, Apple no autoriza las animaciones Flash en sus iPhone e iPad.

¡La solución serán las animaciones en CSS3!

Con este módulo tendremos infinitas posibilidades de creación, en cuanto comience a ser compatible con los navegadores. En los próximos meses aparecerán sin duda numerosos ejemplos en la Web.

La creación de una animación

Una animación tiene lugar en el tiempo. Para gestionar el tiempo utilizaremos las "imágenes clave",*keyframes* en inglés. Una imagen clave o *keyframe* es una etapa de una animación en la que el elemento animado sufre un cambio.

Con la regla `@keyframes` podrá administrar las etapas de su animación. Todas las animaciones deberán tener una regla `@keyframes`.

Usted deberá establecer las diferentes etapas de su animación e indicar los cambios que deberán aplicarse a las diferentes propiedades. Cada etapa podrá ser identificada con un valor en porcentaje, siendo 100 % la duración total de la animación.

Podrá indicar la **duración** de su animación con la propiedad `animation-duration`.

El **movimiento** de la animación estará gestionado por la propiedad `animation-timing-function`. Podrá aplicar aumentos y disminuciones de la velocidad, al igual que con las transiciones.

Del mismo modo podrá indicar cuántas veces deberá repetirse la animación. Se trata de la noción de**iteración**. Usted podrá indicarla con la propiedad `animation-iteration-count`.

La propiedad `animation-delay` sirve para indicar que la animación deberá comenzar tras un **plazo**determinado, y no inmediatamente.

La propiedad `animation-direction` le permite indicar si la animación deberá realizarse en el sentido inverso.

La propiedad `animation-play-state` sirve para indicar si la animación se está ejecutando o si está en pausa.

Podrá usar la sintaxis abreviada `animation` siguiendo este orden:

- `animation-name`
- `animation-duration`
- `animation-timing-function`
- `animation-delay`
- `animation-iteration-count`
- `animation-direction`

Y no olvide que deberá utilizar, claro está, los prefijos propietarios de los navegadores.

Un desplazamiento simple

1. Objetivo

Vamos a desplazar, ininterrumpidamente, un caja `<div>` de izquierda a derecha y luego de derecha a izquierda.

2. La estructura

Vamos a crear una caja `<div>` con el código de identificación `#caja`.

```
<div id="caja"><p> </p></div>
```

Le vamos a aplicar un estilo muy sencillo:

```
#caja {
   background-color: red;
   position: relative;
   width: 100px;
   height: 100px;
}
```

3. La animación

Nuestra caja roja deberá desplazarse hasta una posición fijada en la mitad de la duración total y, a continuación, volverá a su posición inicial, para finalizar la animación. Necesitaremos por lo tanto tres keyframes: a 0%, 50% y 100%. Indicaremos la posición a la que deberá llegar en cada etapa con la propiedad `left`. Hemos llamado a la animación `redBox`.

```
@-webkit-keyframes redBox {
    0% {
        left: 10px;
    }
    50% {
        left: 400px;
    }
    100% {
        left: 10px;
    }
}
@-moz-keyframes redBox {
    0% {
        left: 10px;
    }
    50% {
        left: 400px;
    }
    100% {
        left: 10px;
    }
}
@keyframes 'redBox' {
    0% {
        left: 10px;
    }
    50% {
        left: 400px;
```

```
    }
    100% {
        left: 10px;
    }
}
```

En las propiedades CSS de la caja `<div>`, indicaremos las propiedades necesarias:

- el nombre de la animación: `animation-name`.
- la duración de la animación: `animation-duration`.
- las repeticiones de la animación: `animation-iteration`.

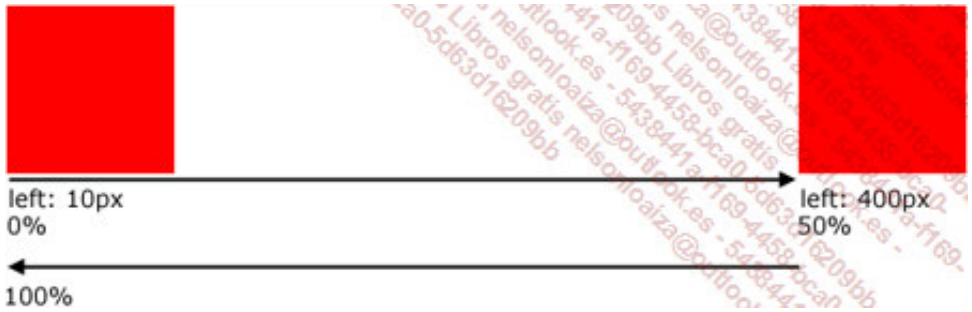
```
#caja {
    background-color: red;
    position: relative;
    width: 100px;
    height: 100px;
    -webkit-animation-name: redBox;
    -webkit-animation-duration: 5s;
    -webkit-animation-iteration-count: infinite;
    -moz-animation-name: redBox;
    -moz-animation-duration: 5s;
    -moz-animation-iteration-count: infinite;
    animation-name: redBox;
    animation-duration: 5s;
    animation-iteration-count: infinite;
}
```

O bien, con la sintaxis abreviada:

```
#caja {
    background-color: red;
    position: relative;
    width: 100px;
    height: 100px;
    -webkit-animation: redBox 5s infinite;
    -moz-animation: redBox 5s infinite;
    animation: redBox 5s infinite;
}
```

4. La visualización

Así se verá en la pantalla:



Rotación y transparencia

1. Objetivo

Vamos a crear una animación en la cual una imagen va a girar sobre sí misma y se va a volver transparente.

2. La estructura

Cree una caja `<div>` con un código de identificación.

```
<div id="imagen"><img src="leon.jpg" alt="León" /></div>
```

Aplíquele una posición absoluta.

```
#imagen {
    position: absolute;
    left: 400px;
    top:  300px;
}
```

3. La animación

Para hacer que la imagen se vuelva transparente vamos a usar la propiedad `opacity`.

Para hacer que la imagen gire sobre ella misma vamos a usar la propiedad `transform`.

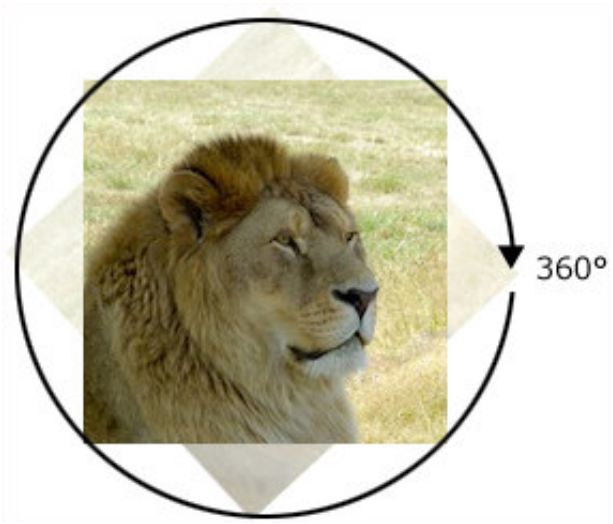
```
@-webkit-keyframes girar {
        0% {
        opacity: 1;
        -webkit-transform: rotate(0deg);
    }
    100% {
        opacity: 0;
        -webkit-transform: rotate(360deg);
    }
}
@-moz-keyframes girar {
    0% {
        opacity: 1;
        -moz-transform: rotate(0deg);
    }
    100% {
        opacity: 0;
        -moz-transform: rotate(360deg);
    }
}
@keyframes girar {
    0% {
        opacity: 1;
        transform: rotate(0deg);
    }
    100% {
        opacity: 0;
        transform: rotate(360deg);
    }
}
```

Ahora vamos a aplicar esta animación con el selector de la caja:

```
#imagen {
    position: absolute;
    left: 400px;
    top:  300px;
    -webkit-animation: girar 5s infinite;
    -moz-animation: girar 5s infinite;
    animation: girar 5s infinite;
}
```

4. La visualización

Este es el resultado final esperado:



Un formulario que parpadee

1. Objetivo

Queremos resaltar un formulario en un página web haciendo que su borde parpadee. No es muy "estético", ¡pero se trata de un ejemplo!

2. La estructura del formulario

Vamos a crear un formulario normal y corriente:

```
<form id="registro" method="#" action="#">
    <p>
       <label for="nombre">Su nombre: </label>
       <input type="text" id="nombre" />
    </p>
    <p>
       <label for="email">Su e-mail: </label>
       <input type="email" id="email" />
    </p>
    <p>
       <input type="submit" id="enviar" value="Enviar" />
    </p>
</form>
```

3. El diseño

Le aplicaremos un diseño al formulario con los estilo CSS3 habituales:

```
#registro{
   width: 300px;
   padding: 10px;
   border: 1px solid #333;
   border-radius: 10px;
}
```

4. La animación

Para hacer que el borde del formulario parpadee, vamos a aplicarle una sombra que pasará de rojo a amarillo.

Vamos a crear una animación a la que llamaremos `sombra`. En la primera etapa, en el 0%, definimos una sombra roja de 15 píxeles y, en el 100%, una sombra amarilla.

```
@-webkit-keyframes sombra {
    0% {
        box-shadow: 0 0 15px red;
    }
    100% {
        box-shadow: 0 0 15px yellow;
    }
}
@-moz-keyframes sombra {
    0% {
        box-shadow: 0 0 15px red;
```

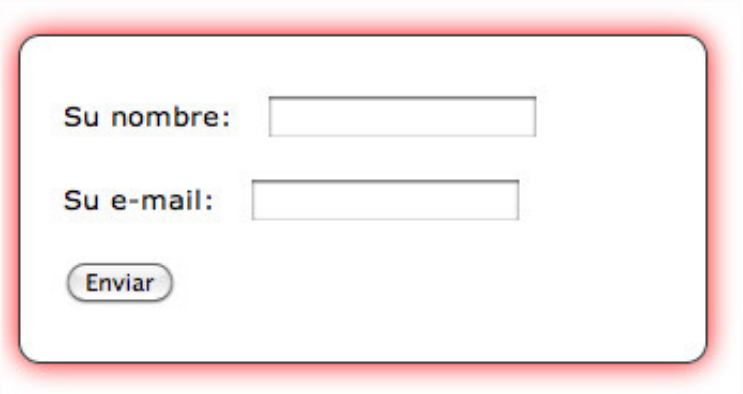
```
    }
    100% {
        box-shadow: 0 0 15px yellow;
    }
}
@keyframes sombra {
    0% {
        box-shadow: 0 0 15px red;
    }
    100% {
        box-shadow: 0 0 15px yellow;
    }
}
```

Vamos a aplicar esta animación al formulario. La animación tendrá un movimiento `ease-in` (se acelerará al final), no tendrá fin (`infinite`), alternará el sentido de la animación (`alternate`) y durará 500 milisegundos (`500ms`).

```
#registro {
    width: 300px;
    padding: 10px;
    border: 1px solid #333;
    border-radius: 10px;
    /* Animación */
    -webkit-animation: sombra ease-in infinite alternate 500ms;
    -moz-animation: sombra ease-in infinite alternate 500ms;
    animation: sombra ease-in infinite alternate 500ms;
}
```

5. La visualización

Este será el resultado obtenido: ¡image que el borde parpadea y que le escuecen los ojos de verlo!



Una ventana modal

1. Objetivo

Vamos a crear lo que se conoce como "ventana modal". Una ventana modal es una ventana que se abre en la página web por encima del contenido en curso y que luego podemos cerrar haciendo clic con el ratón.

Hemos sacado este ejemplo del sitio web de Paul Hayes: http://www.paulrhayes.com/2011-03/css-modal/

2. El botón

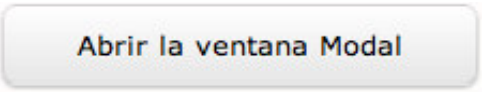
La ventana modal aparecerá cuando se haga clic en un botón. En primer lugar, vamos a crear un vínculo `a` en la página HTML para el botón. Este vínculo usará la clase que llamaremos `openModal`.

```
<p><a href="#ejemplo" class="openModal">Abrir la ventana Modal</a></p>
```

Ahora podrá aplicarle el formato que quiera. Le proponemos el siguiente:

```
a.openModal {
    margin: 1em auto;
    display: block;
    width: 200px;
    background: #ccc;
    text-align: center;
    text-decoration: none;
    color: black;
    padding: 10px;
    -moz-border-radius: 7px;
    border-radius: 7px;
    background: -moz-linear-gradient(#fff, #ddd);
    background: -webkit-gradient(linear, right top, right bottom,
from(rgb(255,255,255)), to(rgb(230,230,230)));
    text-shadow: 0 1px 0 #fff;
    border: 1px solid rgba(0,0,0,0.1);
    -webkit-box-shadow: 0 1px 1px rgba(0,0,0,0.3);
    -moz-box-shadow: 0 1px 1px rgba(0,0,0,0.3);
    box-shadow: 0 1px 1px rgba(0,0,0,0.3);
}
```

Así se visualizará el botón:



3. La ventana modal

La ventana modal es un elemento `aside` que presenta el código de identificación `ejemplo` y que usa la clase `modal`. El código de identificación es importante, ya que lo vamos a usar con la pseudo-clase `target`.

El título del contenido es un elemento `h2` y el texto del contenido es un elemento `p`. El botón de cierre es un vínculo que dispone de su propio código de identificación `close` y de un `title` con la

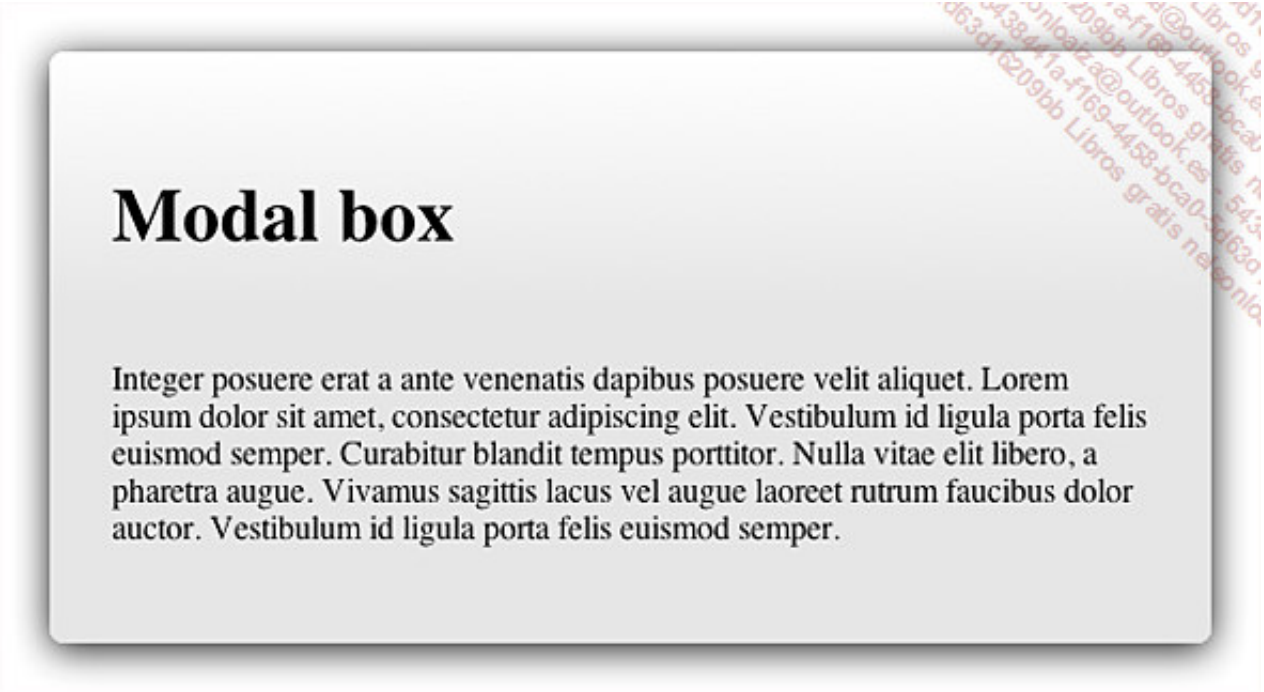
etiqueta `close`.

```
<aside id="ejemplo" class="modal">
    <div>
        <h2>Modal box</h2>
        <p>Integer posuere...</p>
        <a href="#close" title="Close">Cerrar</a>
    </div>
</aside>
```

Estos son los estilos CSS que propone el autor. Se han usado las siguientes propiedades CSS3: bordes redondeados (`border-radius`), una sombra (`box-shadow`), un degradado (`linear-gradient`) y un texto sombreado (`text-shadow`).

```
.modal > div {
    width: 500px;
    background: #fff;
    position: relative;
    margin: 10% auto;
    /* Prettify */
    padding: 30px;
    -moz-border-radius: 7px;
    border-radius: 7px;
    -webkit-box-shadow: 0 3px 20px rgba(0,0,0,0.9);
    -moz-box-shadow: 0 3px 20px rgba(0,0,0,0.9);
    box-shadow: 0 3px 20px rgba(0,0,0,0.9);
    background: -moz-linear-gradient(#fff, #ccc);
    background: -webkit-gradient(linear, right bottom, right
top, color-stop(1, rgb(255,255,255)), color-stop(0.57,
rgb(230,230,230)));
    text-shadow: 0 1px 0 #fff;
}
```

Así se visualizará la ventana con ese diseño:



4. El botón de cierre

El botón de cierre es un vínculo `a` que se encuentra dentro de la caja `<div>` de la ventana modal:

```
<a href="#close" title="Close">Cerrar</a>
```

Sin embargo, nosotros no queremos que se vea la etiqueta del vínculo, **Cerrar,** en el ejemplo. Para eso, el autor aplica una transparencia al color del texto de dicho vínculo, que se encuentra posicionado en posición absoluta, en el borde superior derecho de su elemento padre, la ventana modal:

```
.modal a[href="#close"] {
    position: absolute;
    right: 0;
    top: 0;
    color: transparent;
}
```

En lugar del texto, queremos mostrar una aspa. El autor utiliza la pseudo-clase `:after` y la propiedad `content` para insertar una **X**. Ese elemento presentará una posición absoluta, en el borde superior derecho de la ventana. El círculo ha sido creado con bordes redondeados. Estos son los estilos CSS que propone el autor:

```
.modal a[href="#close"]:after {
    content: 'X';
    display: block;
    /* Position */
    position: absolute;
    right: -10px;
    top: -10px;
    width: 1.5em;
    padding: 1px 1px 1px 2px;
    /* Style */
    text-decoration: none;
    text-shadow: none;
    text-align: center;
    font-weight: bold;
    background: #000;
    color: #fff;
    border: 3px solid #fff;
    -moz-border-radius: 20px;
    border-radius: 20px;
    -webkit-box-shadow: 0 1px 3px rgba(0,0,0,0.5);
    -moz-box-shadow: 0 1px 3px rgba(0,0,0,0.5);
    box-shadow: 0 1px 3px rgba(0,0,0,0.5);
}
```

Este es el botón que habremos creado:

El autor también aplica un estilo al botón de cierre de la ventana cuando se pasa el cursor por encima (`:hover`) y cuando está activo (`:focus`), con una transformación de cambio de escala (`scale`). El botón se ampliará ligeramente y aparecerá subrayado:

```
.modal a[href="#close"]:focus:after,
.modal a[href="#close"]:hover:after {
    -webkit-transform: scale(1.1,1.1);
    -moz-transform: scale(1.1,1.1);
    transform: scale(1.1,1.1);
}
```

```
.modal a[href="#close"]:focus:after {
    outline: 1px solid #000;
}
```

5. Ocultar el cuerpo de la página

Otra característica de esta ventana modal es que "oscurece" la ventana del navegador, es decir, el cuerpo (`body`) de la página web. Ese efecto se obtiene con el elemento `aside`. Dicho elemento utiliza la clase `modal`.

```
<aside id="ejemplo" class="modal">
    ...
</aside>
```

El estilo de la clase `modal` define el posicionamiento (`position: fixed`), aplica un fondo negro semitransparente (`background: rgba(0,0,0,0.5)`) y, cuando se carga la página, hace que la opacidad (`opacity`) sea de `0`, de modo que esta no aparecerá "oscurecida". Por último, cuando se use la ventana modal, el cuerpo de la página "se ocultará" progresivamente. Esto se consigue con la propiedad `transition` sobre la opacidad (`opacity`).

```
.modal {
    /* Overlay page content */
    position: fixed;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    background: rgba(0,0,0,0.5);
    z-index: 10000;
    /* Hide for now */
    opacity: 0;
    pointer-events: none;
    /* Transformación de la transparencia al abrir la ventana */
    -webkit-transition: opacity 500ms ease-in;
    -moz-transition: opacity 500ms ease-in;
    transition: opacity 500ms ease-in;
}
```

6. La animación de apertura

Cuando el usuario haga clic en el botón, la ventana modal se abrirá con una animación: se visualizará progresivamente, al mismo tiempo que el fondo oscuro aparecerá también progresivamente.

La animación utilizada es un cambio de escala 3D que solamente funcionará con el motor de renderizado WebKit (Safari y Chrome). Mozilla Firefox ignorará ese efecto y mostrará directamente la ventana modal, sin el efecto de cambio de escala. Además, se le ha aplicado una sombra. El autor ha previsto cuatro etapas para esta animación.

Este es el código CSS de la animación, a la que se le ha dado el nombre de `bounce` :

```
@-webkit-keyframes bounce {
    0% {
        -webkit-transform: scale3d(0.1,0.1,1);
        -webkit-box-shadow: 0 3px 20px rgba(0,0,0,0.9);
        -moz-box-shadow: 0 3px 20px rgba(0,0,0,0.9);
        transform: scale3d(0.1,0.1,1);
        box-shadow: 0 3px 20px rgba(0,0,0,0.9);
    }
    55% {
        -webkit-transform: scale3d(1.08,1.08,1);
        -webkit-box-shadow: 0 10px 20px rgba(0,0,0,0);
        -moz-box-shadow: 0 10px 20px rgba(0,0,0,0);
        transform: scale3d(1.08,1.08,1);
        box-shadow: 0 10px 20px rgba(0,0,0,0);
    }
    75% {
        -webkit-transform: scale3d(0.95,0.95,1);
        -webkit-box-shadow: 0 0 20px rgba(0,0,0,0.9);
        -moz-box-shadow: 0 0 20px rgba(0,0,0,0.9);
        transform: scale3d(0.95,0.95,1);
        box-shadow: 0 0 20px rgba(0,0,0,0.9);
    }
    100% {
        -webkit-transform: scale3d(1,1,1);
        -webkit-box-shadow: 0 3px 20px rgba(0,0,0,0.9);
        -moz-box-shadow: 0 3px 20px rgba(0,0,0,0.9);
        transform: scale3d(1,1,1);
        box-shadow: 0 3px 20px rgba(0,0,0,0.9);
    }
}
```

7. Activar la animación de apertura

Esta animación deberá aplicarse, en el momento de ocultar el cuerpo de la página (`.modal:target`), sobre la ventana modal que ha sido insertada como primer elemento hijo (`.modal:target > div`).

```
.modal:target > div {
    -webkit-animation-name: bounce;
    -moz-animation-name: bounce;
    animation-name: bounce;
}
```

8. Aplicar la ocultación

Ahora vamos a aplicar el efecto que permitirá ocultar la página:

```
.modal:target {
 opacity: 1;
 pointer-events: auto;
}
```

9. La animación de cierre

La animación de cierre aplica un cambio de escala a la ventana modal, para visualizarla con una proporción de `0.1`.

```
@-webkit-keyframes minimise {
    0% {
        -webkit-transform: scale3d(1,1,1);
        transform: scale3d(1,1,1);
    }
    100% {
        -webkit-transform: scale3d(0.1,0.1,1);
        transform: scale3d(0.1,0.1,1);
    }
}
```

10. Activar la animación de cierre

Esta animación deberá aplicarse cuando se oculte la ventana modal, que ha sido insertada como primer elemento hijo (`.modal > div`).

```
.modal > div {
    ...
    /* Aplicación de la animación de cierre */
    -webkit-animation: minimise 500ms linear;
    -moz-animation: minimise 500ms linear;
    animation: minimise 500ms linear;
    ...
}
```

11. El código completo

Este es el código completo de la página web:

```
<!DOCTYPE HTML>
<html lang="es">
<head>
<title>Título del documento</title>
<meta charset="UTF-8" />
<style>
/* El contenedor que oculta el body */
.modal {
     /* Overlay page content */
     position: fixed;
     top: 0;
     left: 0;
     right: 0;
     bottom: 0;
     background: rgba(0,0,0,0.5);
     z-index: 10000;
     /* Transformación de la transparencia al abrir la página */
     -webkit-transition: opacity 500ms ease-in;
     -moz-transition: opacity 500ms ease-in;
     transition: opacity 500ms ease-in;
     /* Hide for now */
     opacity: 0;
     pointer-events: none;
}
/* La caja div del contenido de la ventana modal */
.modal > div {
     width: 500px;
     background: #fff;
     position: relative;
     margin: 10% auto;
     /* Aplicación de la animación de cierre */
```

```
     -webkit-animation: minimise 500ms linear;
     -moz-animation: minimise 500ms linear;
     animation: minimise 500ms linear;
     /* Prettify */
     padding: 30px;
     -moz-border-radius: 7px;
     border-radius: 7px;
     -webkit-box-shadow: 0 3px 20px rgba(0,0,0,0.9);
     -moz-box-shadow: 0 3px 20px rgba(0,0,0,0.9);
     box-shadow: 0 3px 20px rgba(0,0,0,0.9);
     background: -moz-linear-gradient(#fff, #ccc);
     background: -webkit-gradient(linear, right bottom, right top,
color-stop(1, rgb(255,255,255)), color-stop(0.57,
rgb(230,230,230)));
     text-shadow: 0 1px 0 #fff;
}
/* El botón de apertura */
a.openModal {
     margin: 1em auto;
     display: block;
     width: 200px;
     background: #ccc;
     text-align: center;
     text-decoration: none;
     color: black;
     padding: 10px;
     -moz-border-radius: 7px;
     border-radius: 7px;
     background: -moz-linear-gradient(#fff, #ddd);
     background: -webkit-gradient(linear, right top, right bottom,
from(rgb(255,255,255)), to(rgb(230,230,230)));
     text-shadow: 0 1px 0 #fff;
     border: 1px solid rgba(0,0,0,0.1);
     -webkit-box-shadow: 0 1px 1px rgba(0,0,0,0.3);
     -moz-box-shadow: 0 1px 1px rgba(0,0,0,0.3);
     box-shadow: 0 1px 1px rgba(0,0,0,0.3);
}

/* El vínculo-botón de cierre */
.modal a[href="#close"] {
     position: absolute;
     right: 0;
     top: 0;
     color: transparent;
}
/*  El vínculo-botón de cierre */
.modal a[href="#close"]:after {
     content: ’X’;
     display: block;
     /* Posicionamiento */
     position: absolute;
     right: -10px;
     top: -10px;
     width: 1.5em;
     padding: 1px 1px 1px 2px;
     /* Formato */
     text-decoration: none;
     text-shadow: none;
     text-align: center;
     font-weight: bold;
     background: #000;
     color: #fff;
     border: 3px solid #fff;
     -moz-border-radius: 20px;
     border-radius: 20px;
```

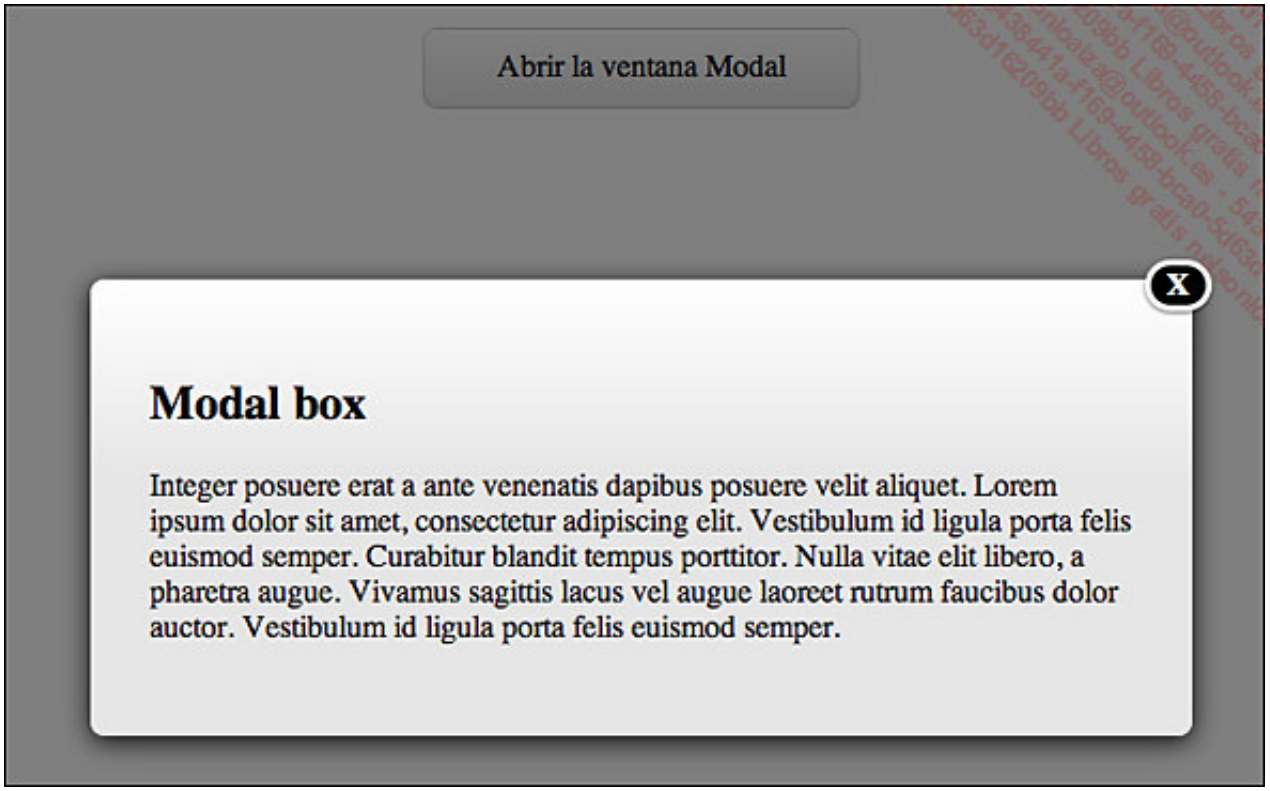
```
     -webkit-box-shadow: 0 1px 3px rgba(0,0,0,0.5);
     -moz-box-shadow: 0 1px 3px rgba(0,0,0,0.5);
     box-shadow: 0 1px 3px rgba(0,0,0,0.5);
}
/* El vínculo-botón de cierre cuando se pase el cursor o cuando esté
activo */
.modal a[href="#close"]:focus:after,
.modal a[href="#close"]:hover:after {
     -webkit-transform: scale(1.1,1.1);
     -moz-transform: scale(1.1,1.1);
     transform: scale(1.1,1.1);
}
.modal a[href="#close"]:focus:after {
     outline: 1px solid #000;
}
/* Ocultar la página  */
.modal:target {
     opacity: 1;
     pointer-events: auto;
}

/* Aplicación de la animación de apertura */
.modal:target > div {
     -webkit-animation-name: bounce;
     -moz-animation-name: bounce;
     animation-name: bounce;
}

/* Animación de apertura  */
@-webkit-keyframes bounce {
   0% {
     -webkit-transform: scale3d(0.1,0.1,1);
     -webkit-box-shadow: 0 3px 20px rgba(0,0,0,0.9);
     -moz-box-shadow: 0 3px 20px rgba(0,0,0,0.9);
     transform: scale3d(0.1,0.1,1);
     box-shadow: 0 3px 20px rgba(0,0,0,0.9);
   }
   55% {
     -webkit-transform: scale3d(1.08,1.08,1);
     -webkit-box-shadow: 0 10px 20px rgba(0,0,0,0);
     -moz-box-shadow: 0 10px 20px rgba(0,0,0,0);
     transform: scale3d(1.08,1.08,1);
     box-shadow: 0 10px 20px rgba(0,0,0,0);
   }
   75% {
     -webkit-transform: scale3d(0.95,0.95,1);
     -webkit-box-shadow: 0 0 20px rgba(0,0,0,0.9);
     -moz-box-shadow: 0 0 20px rgba(0,0,0,0.9);
     transform: scale3d(0.95,0.95,1);
     box-shadow: 0 0 20px rgba(0,0,0,0.9);
   }
   100% {
     -webkit-transform: scale3d(1,1,1);
     -webkit-box-shadow: 0 3px 20px rgba(0,0,0,0.9);
     -moz-box-shadow: 0 3px 20px rgba(0,0,0,0.9);
     transform: scale3d(1,1,1);
     box-shadow: 0 3px 20px rgba(0,0,0,0.9);
   }
}

/* Animación de cierre */
@-webkit-keyframes minimise {
   0% {
     -webkit-transform: scale3d(1,1,1);
     transform: scale3d(1,1,1);
```

```
    }
    100% {
      -webkit-transform: scale3d(0.1,0.1,1);
      transform: scale3d(0.1,0.1,1);
    }
}
</style>
</head>
<body>
<p><a href="#ejemplo" class="openModal">Abrir la ventana
Modal</a></p>
<aside id="ejemplo" class="modal">
   <div>
     <h2>Modal box</h2>
     <p>Integer posuere erat a ante venenatis dapibus
posuere velit aliquet. Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Vestibulum id ligula porta felis euismod
semper. Curabitur blandit tempus porttitor. Nulla vitae elit
libero, a pharetra augue. Vivamus sagittis lacus vel augue
laoreet rutrum faucibus dolor auctor. Vestibulum id ligula
porta felis euismod semper.</p>
     <a href="#close" title="Close">Cerrar</a>
   </div>
</aside>
</body>
</html>
```

12. La visualización de la página

Así se visualizará inicialmente la página:

Cuando haga clic en el botón, el fondo oscuro y la ventana modal aparecerán progresivamente y ya no se podrá usar el botón de apertura:



Cuando haga clic en el botón de cierre de la ventana modal, se producirá el efecto contrario.

Situación actual

En la actualidad (y desde hace ya bastante tiempo) insertar audio y vídeo en las páginas web es muy común, banal, y la mayoría de los internautas ya no les prestan atención. Pero, técnicamente, es otra historia. Como usted sabe, los navegadores web solo saben interpretar y, por lo tanto, mostrar, el texto, las imágenes estáticas (con los formatos .gif, .jpg y .png), las imágenes animadas (con formato .gif animado) y JavaScript. Si los sacamos de ahí, los navegadores se vuelven sordomudos. Para todo lo que sea contenido multimedia e interactivo será necesario añadir obligatoriamente un motor de interpretación en el navegador, para que pueda publicar aquellos elementos que no sepa interpretar de forma nativa. Se trata de los famosos **plugins**. El gran vencedor fue Flash, una tecnología propietaria de Adobe que permite insertar animaciones interactivas, así como audio y vídeo, con un único plugin: **FlashPlayer**.

HTML5 ha revolucionado el panorama al introducir los elementos `audio` y `vídeo`. Esos dos nuevos elementos permiten insertar archivos multimedia "casi" de forma nativa, sin necesidad de añadir plugins propietarios.

Evitar un plugin siempre es de agradecer. Los plugins son siempre una fuente de problemas: código de inserción HTML pesado, inestabilidad, brechas de seguridad, problemas con la versión, bloqueo de la instalación...

Sin embargo, la revolución del HTML5 no va a acabar con todo lo que ya existe de la noche a la mañana. Todavía nos encontramos con muchos problemas a la hora de usar elementos multimedia en HTML5: problemas de formato de audio y vídeo, problemas de protección de los originales (DRM), problemas de compatibilidad con los distintos navegadores, interacción con el usuario...

Insertar audio

1. El elemento audio

Insertar un archivo de audio (un simple sonido, una canción, una entrevista, un podcast...) es muy sencillo: para eso tenemos el elemento `audio`.

Su atributo indisociable es `src`, que sirve para indicar la ruta de acceso al archivo de la fuente de audio.

Esta sería la sintaxis básica:

```
<audio src="musica.ogg"></audio>
```

2. Los controles de audio

Ahora, debemos permitir que el usuario pueda controlar el audio: que pueda iniciarlo, ponerlo en pausa o pararlo.

En el elemento `audio`, simplemente habrá que utilizar el atributo boleano `controls` para que se inserte la barra de control nativa del navegador.

```
<audio src="musica.ogg" controls></audio>
```

3. La apariencia del reproductor de audio

Como es habitual, la apariencia del lector de audio dependerá del navegador:

Este es el reproductor de audio de Opera 12:

Este es el reproductor de audio de Chrome 29:

Este es el reproductor de audio de Internet Explorer 10:

Este es el reproductor de audio de Safari 6:

Este es el reproductor de audio de Firefox 23:

4. Los atributos para el audio

Con el elemento `audio` podrá utilizar estos atributos:

- `autoplay`: este atributo boleano inicia la lectura del archivo de audio en cuanto se encuentre disponible, al cargarse la página web. En mi humilde opinión, no es muy acertado utilizarlo, ya que es preferible que sea el propio usuario quien decida si quiere escuchar o no dicho archivo de audio.
- `loop`: este atributo boleano permite reproducir el sonido en bucle. De nuevo, no resulta muy ergonómico.
- `preload`: indica al navegador que deberá cargar el archivo de audio cuando cargue la página web, de modo que esté disponible para el usuario lo antes posible. Este atributo`preload` será ignorado si usa `autoplay`. El atributo `preload` acepta los valores `auto`(equivale a usar el atributo como si fuera boleano, es decir, sin especificar ningún valor),`none` (sin descarga) y `metadata` (para precargar los metadatos asociados al archivo). Tenga cuidado, si tiene que precargar varios archivos de audio, podría consumir demasiado ancho de banda.

Esta es una sintaxis completa (¡y antiergonómica!):

```
<audio src="musica.ogg" controls preload autoplay loop></audio>
```

5. Los formatos de audio

En la sintaxis del ejemplo anterior, he indicado de forma arbitraria un archivo de audio con el formato .ogg. Sin embargo, existen muchos otros formatos de audio, ¡y es ahí donde todo se complica! El W3C no indica de ninguna manera qué formato se debe usar para insertar un archivo de audio en una página web. Así que hay una especie de "guerra" de formatos de audio entre los navegadores.

Esta es la situación en el momento de redactar estas páginas (septiembre de 2013) en cuanto a la compatibilidad de los principales navegadores, en su última versión, con los diferentes formatos. En la siguiente tabla solamente he recogido los formatos de audio para la Web (.ogg, .mp3 y .acc) y he preferido dejar fuera los demás formatos que no se adaptan a la Web (.aiff, .wav, .au...). Debe saber que el formato WebM puede reproducir tanto audio, como vídeo (véase la sección Insertar vídeo - Los formatos de vídeo).

	ogg	**mp3**	**acc**
Internet Explorer	No	Sí	No
Chrome	Sí	Sí	Sí
Firefox	Sí	No	No
Safari	No	Sí	Sí
Opera	Sí	No	No

Como puede ver, la conclusión es "simple": ¡no existe ningún formato de audio que sea compatible "universalmente" con los cinco navegadores principales del mercado!

Aún así, podemos encontrar rápidamente una solución para este problema: bastará con que indiquemos varios archivos de fuentes de audio que sean compatibles con los principales navegadores. Con los formatos .mp3 y .ogg estaremos abarcando los cinco navegadores principales del mercado.

Para indicar varios archivos de fuentes de audio, deberá usar el elemento `source` con su atributo`src`, en el elemento `audio`.

Esta es la sintaxis que debemos usar:

```
<audio>
    <source src="musica.mp3">
    <source src="musica.ogg">
</audio>
```

Los navegadores abrirán y reproducirán el primer archivo de audio que sepan interpretar.

Para el elemento `source`, puede añadir el atributo `type`, que permite indicar el tipo de archivo fuente y mejorar así la gestión de los archivos de audio por parte del navegador:

```
<audio>
    <source src="musica.ogg" type="audio/ogg"/>
    <source src="musica.mp3" type="audio/mpeg"/>
    <source src="musica.acc" type="audio/acc"/>
</audio>
```

En el atributo `type` también puede indicar el codec utilizado:

```
<audio>
    <source src="musica.ogg" type="audio/ogg ; codecs='vorbis '"/>
</audio>
```

6. Diseñe sus propios controles de audio

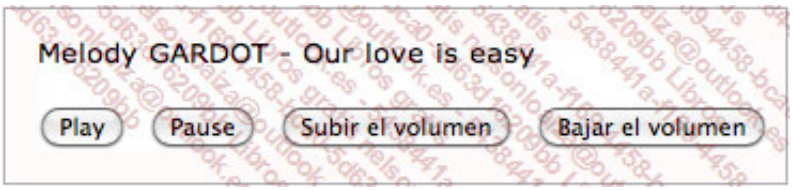
Acabamos de ver que cada navegador web utiliza su propia interfaz para mostrar los botones de control del audio. Usted podrá crear su propia interfaz con la API Audio utilizando JavaScript.

Esta sería la sintaxis:

```
<p>Melody GARDOT - Our love is easy</p>
<audio id="mimusica" src="Our-Love-Is-Easy.mp3"></audio>
<div>
<button onclick="document.getElementById('mimusica').play()">Play</button>
<button onclick="document.getElementById('mimusica').pause()">Pause</button>
<button onclick="document.getElementById('mimusica').volume+=0.1">
Subir el volumen</button>
<button onclick="document.getElementById('mimusica').volume-=0.1">
Bajar el volumen</button>
</div>
```

Claro está, no se ha usado el atributo `controls`, ya que queremos crear nuestros propios botones. El elemento `audio` está identificado de forma exclusiva con el atributo `id="mimusica"`. Vamos a crear botones (`button`) que actúen sobre el elemento `mimusica`, empleando los métodos `play()` y `pause()`, y la propiedad `volume`.

No se le ha aplicado aún ningún estilo CSS, ahora usted podrá "decorarlo". Este es el resultado visual:



Si desea saber más sobre la API Audio, consulte la página web que le dedica el sitio web del W3C:http://www.w3.org/wiki/HTML/Elements/audio

Existe una página similar dedicada al vídeo: http://www.w3.org/wiki/HTML/Elements/video

7. Los antiguos navegadores

Aunque partimos del principio de que usted usa HTML5 sabiendo a lo que se expone en cuanto a la compatibilidad con los navegadores, vamos a indicar un mensaje de advertencia para los "antiguos" navegadores que no sean compatibles con el HTML5. Ese mensaje aparecerá en los antiguos navegadores, los cuales van a ignorar todos los elementos HTML5, y será ignorado por los navegadores compatibles.

```
<audio controls preload>
    <source src="musica.mp3" type="audio/mpeg">
    <source src="musica.ogg" type="audio/ogg">
    Su navegador es demasiado antiguo para interpretar HTML5
</audio>
```

También podría insertar aquí una versión Flash de su archivo de audio.

Insertar vídeo

1. El elemento vídeo

Es igual de sencillo que añadir audio: para insertar un vídeo deberá utilizar el elemento `video` con el atributo `src`.

Esta sería la sintaxis básica:

```
<video src="video.mp4"></video>
```

2. Los atributos para el vídeo

Encontramos los mismos atributos que vimos para el audio, con las mismas funcionalidades:`controls`, `preload`, `loop` y `autoplay`.

```
<video src="video.mp4" controls preload></video>
```

Puede usar los atributos `height` y `width` para especificar el alto y el ancho del vídeo. De este modo evitaremos que sea el navegador quien determine esos parámetros y disminuiremos el tiempo de carga de la página web, al igual que ocurre con la inserción de imágenes.

```
<video src="video.mp4" controls preload width="720" height="576></video>
```

Dispone además del atributo `poster`, que le permite mostrar la imagen que usted elija, en lugar de la primera imagen del vídeo, mientras se espera a que este comience. Resulta práctico para mostrar los "títulos de créditos"... ¡de manera estática, claro está!

```
<video src="video.mp4" controls preload width="720"
height="576" poster="image.jpg"></video>
```

3. La apariencia del reproductor de vídeo

Como es habitual, la apariencia del lector de vídeo dependerá del navegador:

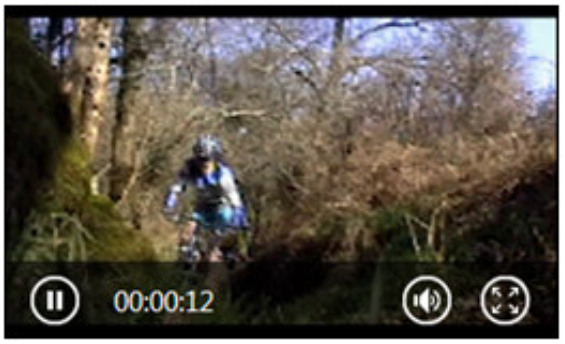
Este es el reproductor de vídeo de Opera 12:

Este es el reproductor de vídeo de Chrome 29:

Este es el reproductor de vídeo de Safari 6:

Este es el reproductor de vídeo de Firefox 23:

Este es el reproductor de vídeo de Internet Explorer 10:



4. Los formatos de vídeo

Para insertar vídeos en una página web, necesitará tres ”capas”, tres ”envoltorios”. La primera es el**formato**, que es solo un ”contenedor”, una caja que albergará el vídeo y el audio. Los formatos se reconocen por sus extensiones: .avi, .mov, .mp4, .mkv, etc. La segunda es el **códec de audio**, que permite comprimir el sonido: MP3, OGG, AAC, etc. La última es el **códec de video**, que permite comprimir las imágenes: H 264 -el más potente pero que aún plantea problemas de derechos y de difusión-, Ogg Theora, muy usado en el ámbito Linux, WebM-, propuesto por Google.

Nuevamente nos encontramos ante una "guerra de formatos". El balance es el mismo que para el audio y las consecuencias son idénticas. Presentamos una tabla en la que se recogen formatos y navegadores en sus últimas versiones (septiembre 2013):

	H.264 (.mp4)	ogg Theora (.ogv)	WebM (.webm)
Internet Explorer	Sí	No	No
Chrome	Sí	Sí	Sí
Firefox	No	Sí	No
Safari	Sí	No	No
Opera	No	Sí	Sí

Al igual que para el audio, es usted quien deberá crear tantos archivos de vídeo como sean necesarios y obtener así la mayor compatibilidad posible con los navegadores más populares. Utilice el nuevo elemento `source` con su atributo indisociable `src`.

Esta sería la sintaxis básica:

```
<video>
    <source src="video.mp4">
    <source src="video.ogg">
    <source src="video.webm">
</video>
```

También puede especificar los formatos y los codecs usados en el vídeo para facilitarle al navegador todos los datos técnicos:

```
<source src="video.ogg" type="video/ogg ;
codecs='theora,vorbis'">
```

5. Convertir vídeos

Para que su vídeo sea compatible con el mayor número posible de navegadores, necesitará un gran número de archivos de vídeo con diversos formatos. Usted puede usar un programa minimalista pero eficaz, gratuito y multiplataforma, se trata de **Miro Video Converter**(http://www.mirovideoconverter.com/). Este programa le permitirá convertir sus vídeos a los formatos .mp4, .webm, .ogv, e incluso a los formatos para iPhone y Android.

La creación de imágenes bitmap con JavaScript

HTML5 introduce un nuevo elemento para el diseño en formato bitmap, se trata del elemento`canvas`. Este elemento permite definir una zona de visualización en la que usted podrá "dibujar", crear animaciones y desarrollar aplicaciones interactivas. En cierta medida, esto va a permitir que los programadores web ya no tengan que usar la tecnología Adobe Flash. Ya no será necesario ningún plugin.

La utilización del elemento `canvas` se dirige, sobre todo, a los programadores que usen JavaScript. En efecto, todo lo que usted podrá crear en el elemento `canvas` se hará con JavaScript. Este libro está orientado al diseño web, y no a la programación en JavaScript, así que solamente vamos a ver los elementos básicos.

Último aviso: debe saber que todo lo que haya sido creado con programación, será siempre más ligero que si se hubiera hecho en diseño "puro" y exportado a los formatos .png, .jpg o .gif. Además, una de las funcionalidades principales de canvas es que ¡el usuario podrá modificar su contenido!

Las técnicas propuestas

Con JavaScript, usted tiene a disposición todo un arsenal de herramientas y de técnicas:

- herramientas de diseño: óvalos, rectángulos, líneas, arcos, curvas de Bézier...
- efectos: bordes, espaciados, degradados, transparencias, sombras...
- transformaciones: rotación, desplazamiento, zoom...
- gestión de la tipografía: incorporación de fuentes tipográficas...
- gestión de las imágenes: importación y exportación.

Dibujar un rectángulo

1. El elemento canvas

Es este elemento `<canvas>` el que le permitirá definir la zona de dibujo en su página web. Deberá atribuirle un `id` único y un tamaño:

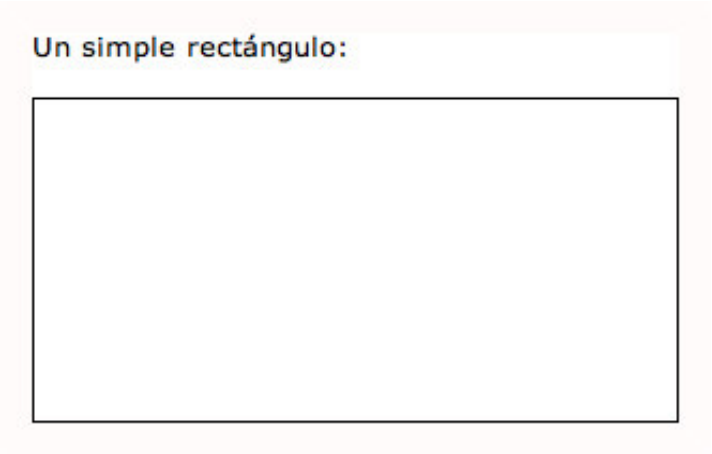
```
<canvas id="rectangulo" width="300" height="150"></canvas>
```

De manera predeterminada, la zona definida no tendrá contorno, así que será invisible. En el`<head>`, en un elemento `<style>`, utilice la propiedad CSS `border` para visualizar su contorno:

```
#rectangulo {
    border: solid 1px #000;
}
```

Por supuesto, también puede usar las propiedades `width` y `height` en el estilo CSS.

Así se visualizará, por el momento, vacío:



2. El rectángulo

Ahora vamos a crear un rectángulo en JavaScript. Inserte el código dentro de un elemento`<script>`, justo delante de `</body>`.

```
<script>
    var miCanvas=document.getElementById("rectangulo");
    var miDibujo=miCanvas.getContext("2d");
    miDibujo.rect(10,10,200,100);
    miDibujo.fill();
</script>
```

La primera línea declara la variable `var` llamada `miCanvas`, que define un nuevo objeto de diseño en el elemento canvas del documento en curso, que ha sido identificado a su vez mediante su `id`exclusivo `rectangulo` (`document.getElementById("rectangulo")`).

La segunda línea declara una nueva variable de objeto llamada `miDibujo`. Este objeto está definido en el elemento miCanvas que acabamos de crear y se usará para crear un gráfico en dos dimensiones (`miCanvas.getContext("2d")`).

La tercera línea permite utilizar el método `rect` para definir un rectángulo en el objeto`miDibujo`. Para definir el rectángulo se utilizan las coordenadas x (horizontal) e y (vertical),

10 píxeles y 10 píxeles, contados a partir del ángulo superior izquierdo del elemento canvas, con un ancho de 200 píxeles y una altura de 100 píxeles (`miDibujo.rect(10,10,200,100)`).

La cuarta línea solicita que se cree o se muestre el rectángulo definido: `miDibujo.fill()`.

También podríamos haber usado el método `fillRect()` que permite definir y dibujar el rectángulo al mismo tiempo:

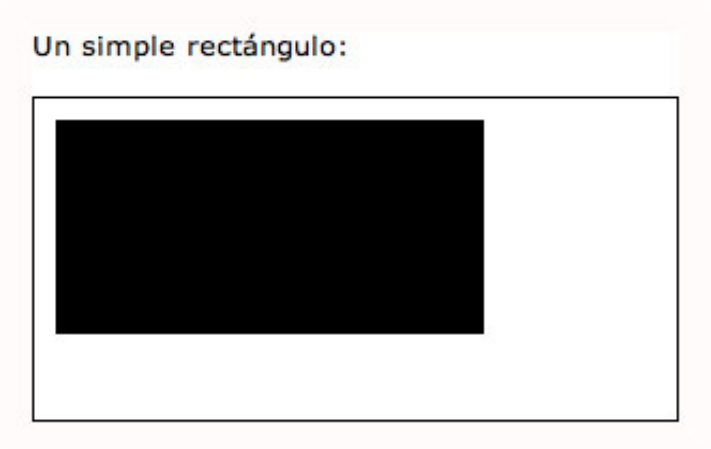
```
<script>
    var miDibujo=document.getElementById("rectangulo").getContext("2d");
    miDibujo.fillRect(10,10,200,100);
</script>
```

3. El código completo y la visualización

Este sería el código completo de la página web:

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Título de la página</title>
    <meta charset="UTF-8" />
    <style>
    #rectangulo {
       border: solid 1px #000;
    }
    </style>
</head>
<body>
<p>Un simple rectángulo:</p>
<canvas id="rectangulo" width="300" height="150"></canvas>
<script>
    var miCanvas=document.getElementById("rectangulo");
    var miDibujo=miCanvas.getContext("2d");
    miDibujo.rect(10,10,200,100);
    miDibujo.fill();
</script>
</body>
</html>
```

La visualización:

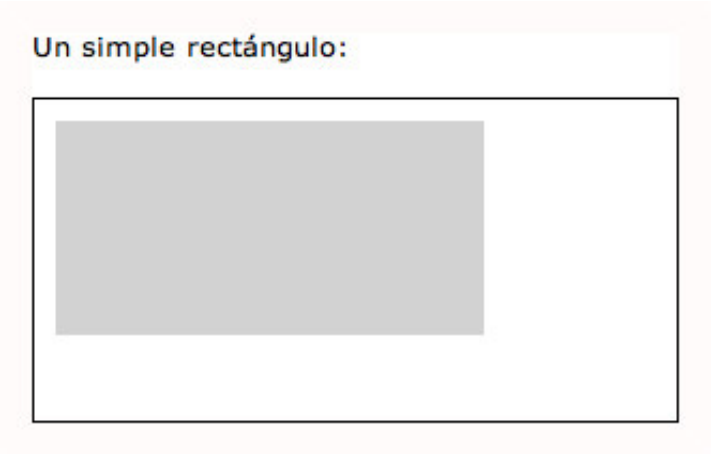


4. Aplicar un color de relleno

Vamos a aplicarle a nuestro objeto de diseño un relleno de color gris claro con el método `fillStyle`. Podemos indicar el color en hexadecimal, en RGB o de forma nominal:

```
var miCanvas=document.getElementById("rectangulo");
var miDibujo=miCanvas.getContext("2d");
miDibujo.fillStyle="rgb(210,210,210)";
miDibujo.rect(10,10,200,100);
miDibujo.fill();
```

Este sería el resultado:



5. Aplicar una sombra

Vamos ahora a aplicarle una sombra a nuestro rectángulo. Para ello, vamos a usar las propiedades:

- `shadowBlur`: para definir el difuminado de la sombra.
- `shadowColor`: para definir el color.
- `shadowOffsetX`: para definir la distancia horizontal (si no se indica nada, la sombra aparecerá alrededor del dibujo).
- `shadowOffsetY`: para definir la distancia vertical (si no se indica nada, la sombra aparecerá alrededor del dibujo).

```
var miCanvas=document.getElementById("rectangulo");
var miDibujo=miCanvas.getContext("2d");
miDibujo.fillStyle="rgb(210,210,210)";
miDibujo.rect(10,10,200,100);
miDibujo.shadowBlur=50;
miDibujo.shadowColor="black";
miDibujo.shadowOffsetX=10;
miDibujo.shadowOffsetY=20;
miDibujo.fill();
```

Este sería el resultado:

Dibujar un círculo

Para dibujar una forma redonda, utilice el método `arc()`. Esta es la sintaxis:`arc(x,y,radius,startAngle,endAngle,antiClockwise)`.

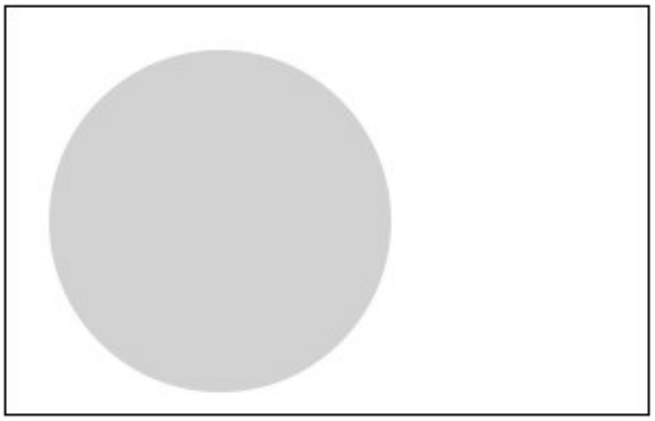
Y sus argumentos:

- `x`: coordenada horizontal calculada a partir del centro del óvalo.
- `y`: coordenada vertical calculada a partir del centro del óvalo.
- `radius`: radio del óvalo.
- `startAngle`: punto de partida del arco en radianes.
- `endAngle`: punto de llegada del arco en radianes.
- `antiClockwise`: valor boleano que indica que el arco será dibujado en el sentido de las agujas del reloj (valor `true`) o al contrario (valor `false`).

Este es el código:

```
<script>
    var miCanvas=document.getElementById("ovalo");
    var miDibujo=miCanvas.getContext("2d");
    miDibujo.fillStyle="rgb(210,210,210)";
    miDibujo.arc(100,100,80,0,2*Math.PI,true);
    miDibujo.fill();
</script>
```

La visualización:



Trazar rayas

Para trazar rayas debemos usar el método `moveTo(x,y)` que determina el punto de partida del trazado. Luego, cada segmento siguiente utilizará el método `lineTo(x,y)`.

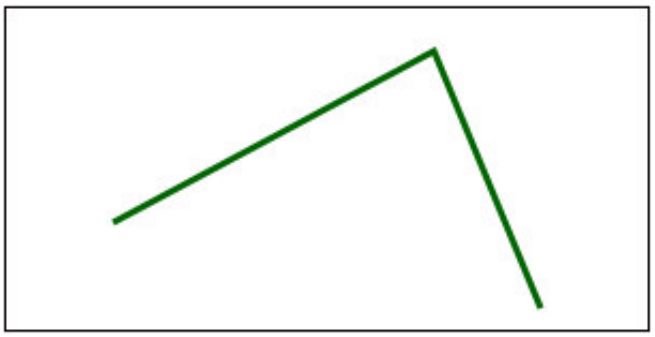
A continuación podrá usar las propiedades `lineWidth` y `strokeStyle` para determinar el grosor y el color de la línea trazada.

Termine de trazar la línea con el método `stroke()`.

Veamos un ejemplo:

```
<script>
    var miCanvas=document.getElementById("trazos");
    var miDibujo=miCanvas.getContext("2d");
    miDibujo.moveTo(50, 100);
    miDibujo.lineTo(200, 20);
    miDibujo.lineTo(250, 140);
    miDibujo.lineWidth=3;
    miDibujo.strokeStyle="darkgreen";
    miDibujo.stroke();
</script>
```

Y el resultado visual:



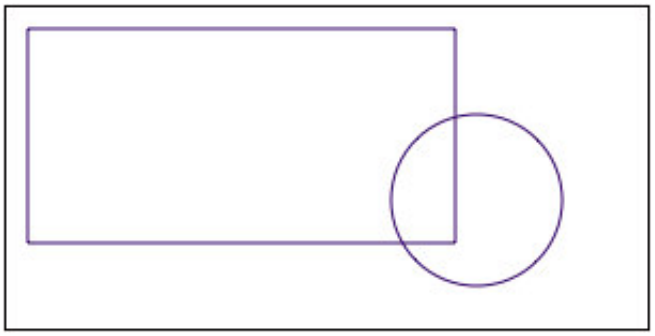
En las técnicas de dibujo, `stroke` indica el borde y `fill` el relleno.

Trazar bordes

La propiedad `strokeStyle` permite definir el color del borde. El método `strokeRect()` permite dibujar el borde de un rectángulo y el método `stroke()` permite dibujar un borde (para un círculo, en el ejemplo siguiente).

```
<canvas id="contorno" width="300" height="150"></canvas>
<script>
    var miCanvas=document.getElementById("contorno");
    var miDibujo=miCanvas.getContext("2d");
    miDibujo.strokeStyle="#306";
    miDibujo.strokeRect(10,10,200,100);
    miDibujo.arc(220,90,40,0,2*Math.PI,true);
    miDibujo.stroke();
</script>
```

Y el resultado visual obtenido:



Añadir texto

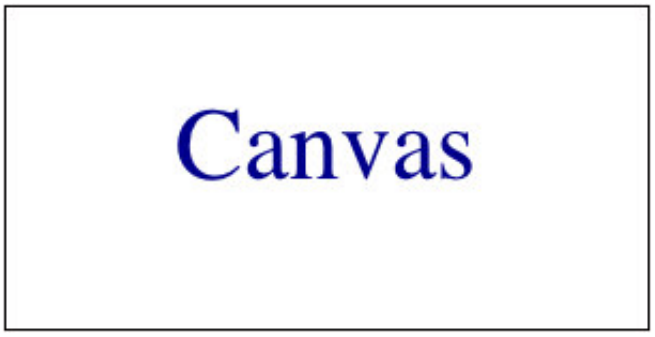
Puede insertar texto en el canvas con el método `fillText()`, para crear un texto de relleno, y`strokeText()`, para crear el borde del texto.

- La propiedad `textBaseline` permite definir la línea de base que prefiramos.
- La propiedad `textAlign` permite definir la alineación del texto.
- La propiedad `font` permite definir la tipografía de caracteres que queramos utilizar.

Este es el código de ejemplo:

```
<!DOCTYPE HTML>
<html lang="es">
<head>
    <title>Texto con <canvas></title>
    <meta charset="UTF-8" />
    <style>
    #texto {
        border: solid 1px #000;
    }
    </style>
</head>
<body>
<canvas id="texto" width="300" height="150"></canvas>
<script>
    var miCanvas=document.getElementById("texto");
    var miTexto=miCanvas.getContext("2d");
    miTexto.textBaseline="alphabetic";
    miTexto.textAlign="center";
    miTexto.font="3em Nueva";
    miTexto.fillStyle="darkblue";
    miTexto.fillText("Canvas",150,80);
</script>
</body>
</html>
```

Y la visualización:



Aplicar un degradado

Puede aplicar degradados lineales o radiales a sus objetos de dibujo.

En primer lugar, deberá crear un objeto degradado lineal (en este caso llamado `dgdl`) en su objeto de dibujo. Los parámetros de este objeto serán las coordenadas de inicio y de fin del degradado. En el siguiente ejemplo, las coordenadas del degradado lineal se han calculado respecto al rectángulo dibujado, para que sea vertical:

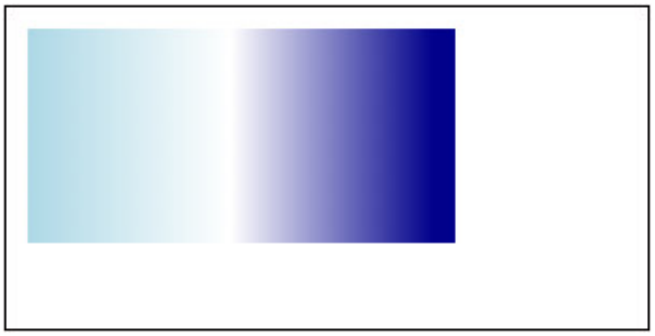
```
var dgdl = miDibujo.createLinearGradient(10, 85, 200, 85);
```

A continuación le aplicamos al degradado el método `addColorStop()` para indicar la posición de inicio del primer color, según una escala que va de 0 (inicio del degradado) a 1 (fin del degradado), así como el color que se usará al inicio. `dgdl.addColorStop(0, "lightblue");`. En el ejemplo, el primer color comienza al principio del degradado (`0`) y se trata de un azul claro (`lightblue`). Haremos lo mismo con el color del medio (`0.5`) y el del final (`1`).

Este es el código:

```
<script>
     var miCanvas=document.getElementById("rectangulo");
     var miDibujo=miCanvas.getContext("2d");
     var dgdl = miDibujo.createLinearGradient(10, 85, 200, 85);
     dgdl.addColorStop(0, "lightblue");
     dgdl.addColorStop(0.5, "#fff");
     dgdl.addColorStop(1, "darkblue");
     miDibujo.fillStyle=dgdl;
     miDibujo.rect(10,10,200,100);
     miDibujo.fill();
</script>
```

Y el resultado obtenido:



Las transformaciones

El elemento `canvas` permite "transformar" los objetos dibujados con los métodos`translate()` (para desplazar un objeto), `rotate(radius)` (para aplicar una rotación a un objeto), y `scale()` (para redimensionar un objeto).

La manipulación de objetos

Usted puede proponer interactividad a sus usuarios con sus objetos dibujados, podría llegar incluso a crear juegos. Pero ahí nos estaríamos alejando de nuestro ámbito de aplicación, el diseño web, y nos estaríamos introduciendo en el mundo de los programadores puros en JavaScript. ¡Y eso es otra historia!

Situación actual y objetivos

Hoy en día, una empresa que desee tener un sitio web "digno de ese nombre" deberá proponer un contenido que se visualice correctamente en todos los medios actuales: en una pantalla de ordenador (¡de 13" a 32"!), en una tableta táctil y en un smartphone.

La dificultad consiste, claro está, en lograr que nuestro contenido sea visible y legible a partir de esos tres tipos de pantalla. El objetivo es proponer una experiencia de uso, una interfaz web, un diseño web, que se adapte automáticamente a las diferentes resoluciones de pantalla que podría utilizar el usuario.

El término inglés que hace referencia a este tipo de arquitectura web es **Responsive Web Design**, que podríamos traducir en la lengua de Cervantes como **Diseño web adaptable o flexible**. Otro término que se suele usar es **Diseño web sensible.**

Para lograr nuestro objetivo, vamos a usar tres técnicas:

- un **grid flexible**, que permita reorganizar la estructura de las páginas, de modo que puedan adaptarse a la resolución de la pantalla.
- **imágenes flexibles**, para que su tamaño se adapte al tamaño de la pantalla de visualización.
- **media queries,** que nos permitan saber cuál es el tipo y las dimensiones del dispositivo del visitante.

Las búsquedas "Media queries"

1. Objetivo

Para crear un diseño web adaptable, será necesario conocer la resolución de la pantalla utilizada, para poder proponer estilos que se adapten a dicha resolución. Ese es precisamente el objeto del módulo **Media Queries**, que se encuentra en fase de **Recommendation** del 19 de junio de 2012:http://www.w3.org/TR/css3-mediaqueries/

Con las búsquedas de medios de difusión, podremos crear hojas de estilo adaptadas a los distintos tamaños de pantalla. La detección se hará automáticamente y el navegador utilizará entonces el estilo adecuado.

2. Con HTML 4 y CSS 2.1

Con HTML 4 y CSS 2.1 podemos usar una hoja de estilo específica para cada medio utilizado. Simplemente tendremos que insertar el atributo `media` en el elemento `link`.

Ejemplo:

```
<link rel="stylesheet" media="screen" href="estiloPantalla.css"
type="text/css" />
<link rel="stylesheet" media="print" href="estiloImpresora.css"
type="text/css" />
```

Podemos usar:

- `screen`: para las pantallas.
- `handheld`: para los dispositivos móviles o de tamaño reducido.
- `print`: para la impresión.
- `aural` (CSS 2.0) / `speech` (CSS 2.1): para la síntesis vocal.
- `braille`: para las páginas en braille.
- `embossed`: para las impresoras en braille.
- `projection`: para el uso en proyectores.
- `tty`: para el uso de terminales.
- `tv`: para la difusión en televisores.
- `all`: para todos los tipos de dispositivos.

Aunque, para la difusión en smartphones, el atributo `handheld` no es compatible con la mayoría de los navegadores, así que de momento no se pueda usar y, encima, no se adapta a las tabletas táctiles.

Debe saber que esta sintaxis HTML 4 / CSS 2.1 se puede usar con una regla `@`:

```
@media screen {
    ...
}
@media print {
    ...
}
```

3. Los criterios de búsqueda de Media Queries en CSS3

El módulo **Media Queries** nos permite seleccionar con toda precisión el medio de difusión, mediante criterios específicos o combinaciones de criterios. El resultado obtenido de esta búsqueda será de tipo boleano: el valor será verdadero o falso. De este modo podremos seleccionar una hoja de estilo en función de las respuestas obtenidas en la búsqueda.

Estos son los criterios que podremos usar en las búsquedas Media Queries:

- El **ancho de la zona de visualización**: `width`. Podemos examinar el ancho de la zona de visualización del navegador. Ejemplo: `width: 780px`.
- La **altura de la zona de visualización**: `height`. Podemos examinar la altura de la zona de visualización en el navegador.
- El **ancho físico**: `device-width`. Podemos examinar el ancho físico de la pantalla de difusión.
- La **altura física**: `device-height`. Podemos examinar la altura física de la pantalla de difusión.
- La **orientación** de la pantalla: `orientation`. Ejemplo: `orientation: portrait` o`orientation: landscape`. Resulta bastante práctico para examinar si el usuario usa su tableta táctil verticalmente (`portrait`) u horizontalmente (`landscape`).
- El **ratio**: `aspect-ratio`. Para examinar el coeficiente ancho/alto. Ejemplo: `aspect-ratio: 16/9`.
- El **ratio físico** : `device-aspect-ratio`. Para examinar el coeficiente físico ancho/alto de la pantalla.
- El **color**: `color`. Podemos examinar si el soporte de difusión utiliza el color (valor por defecto en caso de que no se haya especificado), o el blanco y negro, o una escala de grises. Ejemplo: `min-color: 8`.
- El **número de colores** de la tabla de colores: `color-index`.
- El **número de niveles de gris** para los dispositivos monocromos: `monochrome`.
- La **resolución** de la pantalla de visualización: `resolution`. Expresada en `dpi`.
- El tipo de **exploración** en las pantallas de televisión: `scan`.
- Utilice `grid` para ver si la pantalla de difusión utiliza una cuadrícula con un único tamaño de fuente.

4. La sintaxis de las búsquedas Media Queries

Veámoslo con un ejemplo concreto. Queremos ocuparnos de los periféricos que tengan un ancho exacto de 780 píxeles.

En una página **HTML**, en el `<head>`, insertaremos la búsqueda con el elemento `link`:

```
<link rel="stylesheet" media="screen and (width:780px)"
href="styles780.css" />
```

Esta sería la búsqueda en una página **CSS** con la regla `@`:

```
@media screen and (width:780px) {
    ...
}
```

Indicamos que el tipo de dispositivo es una pantalla: `screen`.

Indicamos que hay un segundo criterio: `and`.

Indicamos que el ancho de la pantalla deberá ser de 780 píxeles: `width: 780px`.

5. Los valores máximos y mínimos

Todos los criterios que acabamos de ver, excepto `orientation`, `scan` y `grid`, admiten los prefijos `min-` y `max-` como valores de criterio.

Por ejemplo, si queremos ocuparnos de las pantallas que tengan un tamaño de 780 píxeles como máximo:

```
@media screen and (max-width:780px) {
    ...
}
```

Este criterio es muy útil, ya que no se suele trabajar para un tamaño fijo de pantalla, ya sea de ordenador, o de otro tipo.

6. Los operadores lógicos

En las búsquedas de medios de difusión podemos usar operadores lógicos para precisar y combinar diferentes criterios.

El operador `and` permite aplicar la lógica **Y**.

Primer ejemplo: queremos examinar las pantallas que presenten un ancho de 780 píxeles y que sean monocromas:

```
@media screen and (width:780px) and monochrome {
    ...
}
```

Segundo ejemplo: queremos examinar las pantallas que tengan una resolución de entre 1024 píxeles, como mínimo, y 1280 píxeles, como máximo:

```
@media screen and (min-width: 1024px) and (max-width: 1280px) {
    ...
}
```

El operador `not` permite utilizar la lógica **NO**.

Ejemplo: queremos examinar las pantallas que no tengan una resolución de 780 píxeles:

```
@media screen and (not width:780px) {
    ...
}
```

La coma `,` permite utilizar la lógica **O**.

Ejemplo: queremos examinar las pantallas que tengan 780 píxeles de ancho o que sean monocromas:

```
@media screen and (width:780px), monochrome {
    ...
}
```

La palabra clave `only` permite indicar que la búsqueda deberá aplicarse **exclusivamente** a los criterios indicados. De este modo podemos ocultar los estilos para los antiguos navegadores:

```
@media only screen and (width:780px) {
   ...
}
```

7. Búsquedas para diferentes soportes

El sitio web http://stuffandnonsense.co.uk/blog/about/hardboiled_css3_media_queries nos propone búsquedas "listas para usar" para los diferentes tipos de pantalla:

```
/* Smartphones (portrait and landscape) ----------- */
@media only screen
and (min-device-width : 320px)
and (max-device-width : 480px) {
/* Styles */
}
/* Smartphones (landscape) ----------- */
@media only screen
and (min-width : 321px) {
/* Styles */
}
/* Smartphones (portrait) ----------- */
@media only screen
and (max-width : 320px) {
/* Styles */
}
/* iPads (portrait and landscape) ----------- */
@media only screen
and (min-device-width : 768px)
and (max-device-width : 1024px) {
/* Styles */
}
/* iPads (landscape) ----------- */
@media only screen
and (min-device-width : 768px)
and (max-device-width : 1024px)
and (orientation : landscape) {
/* Styles */
}
/* iPads (portrait) ----------- */
@media only screen
and (min-device-width : 768px)
and (max-device-width : 1024px)
and (orientation : portrait) {
/* Styles */
}
/* Desktops and laptops ----------- */
@media only screen
and (min-width : 1224px) {
/* Styles */
}
/* Large screens ----------- */
@media only screen
and (min-width : 1824px) {
/* Styles */
}
/* iPhone 4 ----------- */
@media
only screen and (-webkit-min-device-pixel-ratio : 1.5),
only screen and (min-device-pixel-ratio : 1.5) {
/* Styles */
}
```

El tamaño de las pantallas

1. La problemática actual

En la actualidad, cada tipo de smartphone y de tableta presenta sus propias características técnicas en cuanto al tamaño físico de la pantalla y la superficie realmente disponible para la visualización de un sitio web en la versión móvil de un navegador.

El segundo problema está relacionado con lo que se visualizará realmente, en función de las características de la pantalla. Cuando llegó el iPhone de Apple en el 2007, el tamaño de su pantalla se estableció en 320x480 píxeles. Sin embargo, Mobile Safari muestra los sitios web con un ancho de 980 píxeles en los 320 píxeles de ancho que tiene la pantalla, lo que implica una importante disminución de la visualización del sitio web.

2. El tamaño del viewport

Con el iPhone, Apple introdujo un nuevo atributo para el elemento `meta`: `viewport`. Este atributo nos permite controlar el coeficiente del ancho del sitio web (980 píxeles) / ancho visualizado (320 píxeles para Safari en su versión móvil).

Si nosotros queremos que Safari para iPhone muestre un sitio web con un ancho de 320 píxeles (y no de 980 píxeles), tendremos que utilizar el atributo `viewport`:

```
<meta name="viewport" content="width=320" />
```

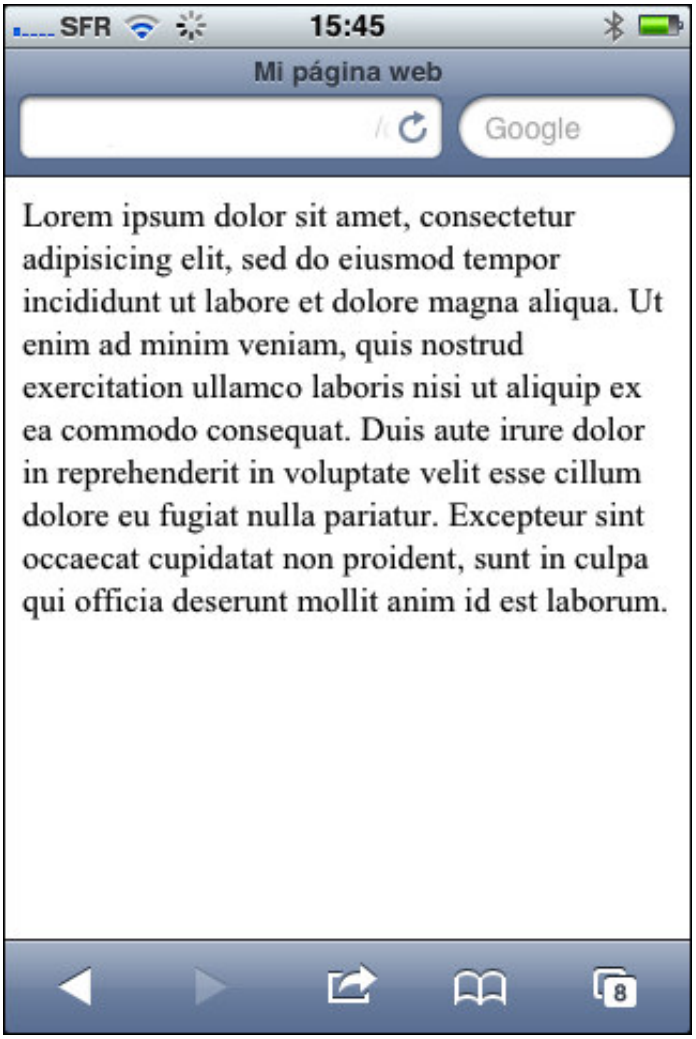
Este atributo ha sido adoptado en la actualidad por todos los constructores de smartphones y tabletas. Se ha convertido de hecho en un estándar.

Veamos un ejemplo simple y concreto para comprender cómo se usa `viewport`. Supongamos que tenemos una página simple que contiene una única caja `<div>` de 980 píxeles de ancho. Así se vería en un iPhone 4: ino se ve nada bien! La visualización de los 980 píxeles del sitio web en los 320 píxeles de la pantalla provocan una disminución importante del contenido visualizado.

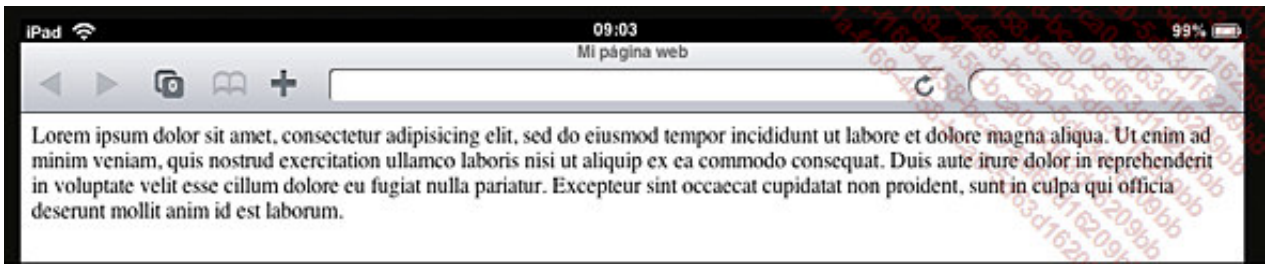
Utilicemos ahora el atributo `viewport` con un valor igual al ancho de visualización de Safari para iPhone:

```
<meta name="viewport" content="width=320" />
```

Este es el resultado obtenido: ahora podemos ver la totalidad de la caja y el tamaño del texto permite la lectura.



Esta sería la misma página web consultada con un iPad (hemos cortado la captura de imagen en la parte inferior):



Todavía tenemos otro problema: el tamaño de las pantallas no está en absoluto harmonizado entre los diferentes smartphones y demás tabletas táctiles. Podremos solucionarlo indicando que se adapte la visualización en función de las características de cada dispositivo. Para eso vamos a utilizar el valor `device-width` que hace referencia al ancho físico de la pantalla.

```
<meta name="viewport" content="width=device-width" />
```

De este modo la visualización se adaptará a la superficie de visualización que proponga cada navegador específico para dispositivos móviles.

3. El zoom de la pantalla

En el `viewport`, en el atributo `content`, podrá insertar otros valores relacionados con el zoom de la pantalla. Esos valores estarán separados de los anteriores mediante una coma.

El valor `initial-scale` especifica el nivel de zoom inicial, cuando se carga la página web. Así, los dispositivos podrán respetar el ancho especificado en las búsquedas de medios de difusión.

```
<meta name="viewport" content="width=device-width,initial-scale=1.0" />
```

El valor `user-scalable` permite definir el nivel de zoom autorizado que puede aplicar el propio usuario. Un valor de 0 impide que el usuario pueda usar el zoom. Un valor de 1 autoriza el uso del zoom.

```
<meta name="viewport" content="width=device-width, user-scalable=0" />
```

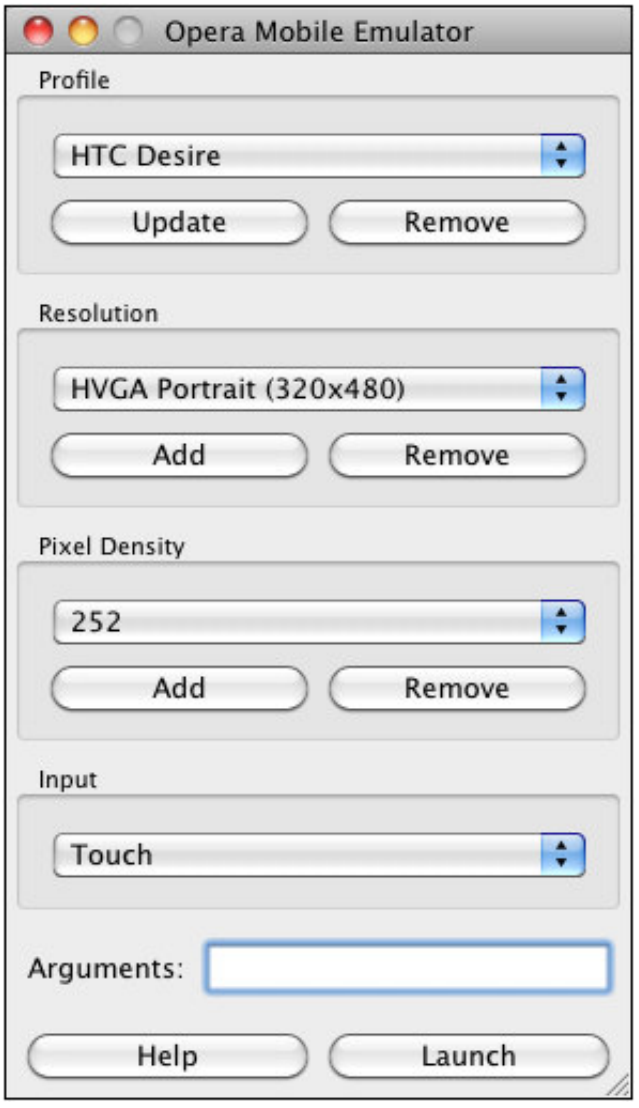
Los valores de `minimum-scale` y `maximum-scale` permiten definir los valores máximos y mínimos de zoom autorizados.

```
<meta name="viewport" content="width=device-width, minimum-scale=0.5,
maximum-scale=3.0, initial-scale=1.0, user-scalable=1" />
```

4. El test de visualización en pantalla

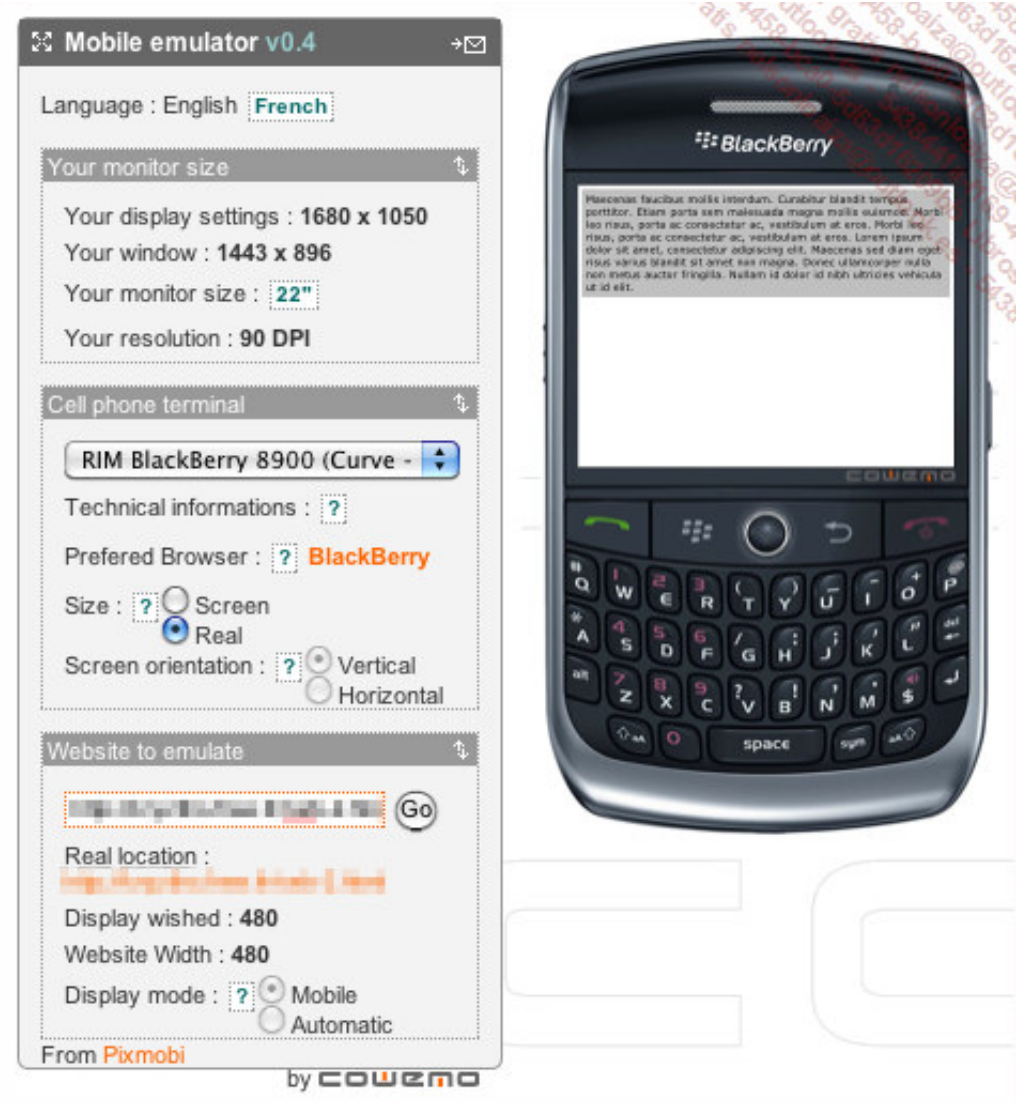
A menos que disponga de todos los modelos de todos los constructores, le resultará muy difícil probar la visualización de un sitio web en todos los smartphones y todas las tabletas que existen.

Sin embargo, puede usar un emulador: **Opera Mobile Emulator**(http://www.opera.com/developer/tools/mobile/). Descargue la versión móvil de Opera y luego seleccione el dispositivo que desee emular.

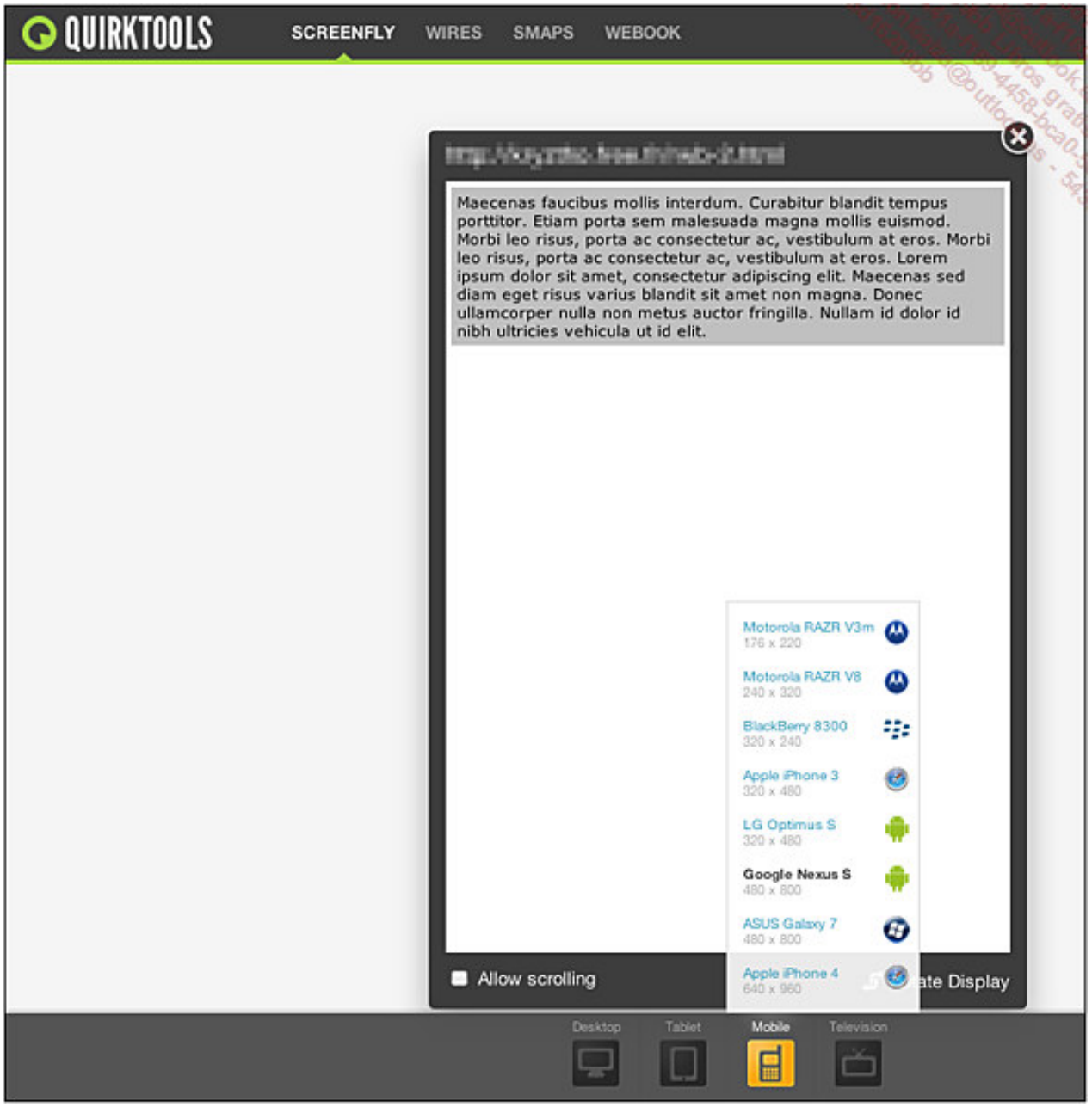


Haga clic en el botón **Launch** y luego inserte la URL que quiera examinar.

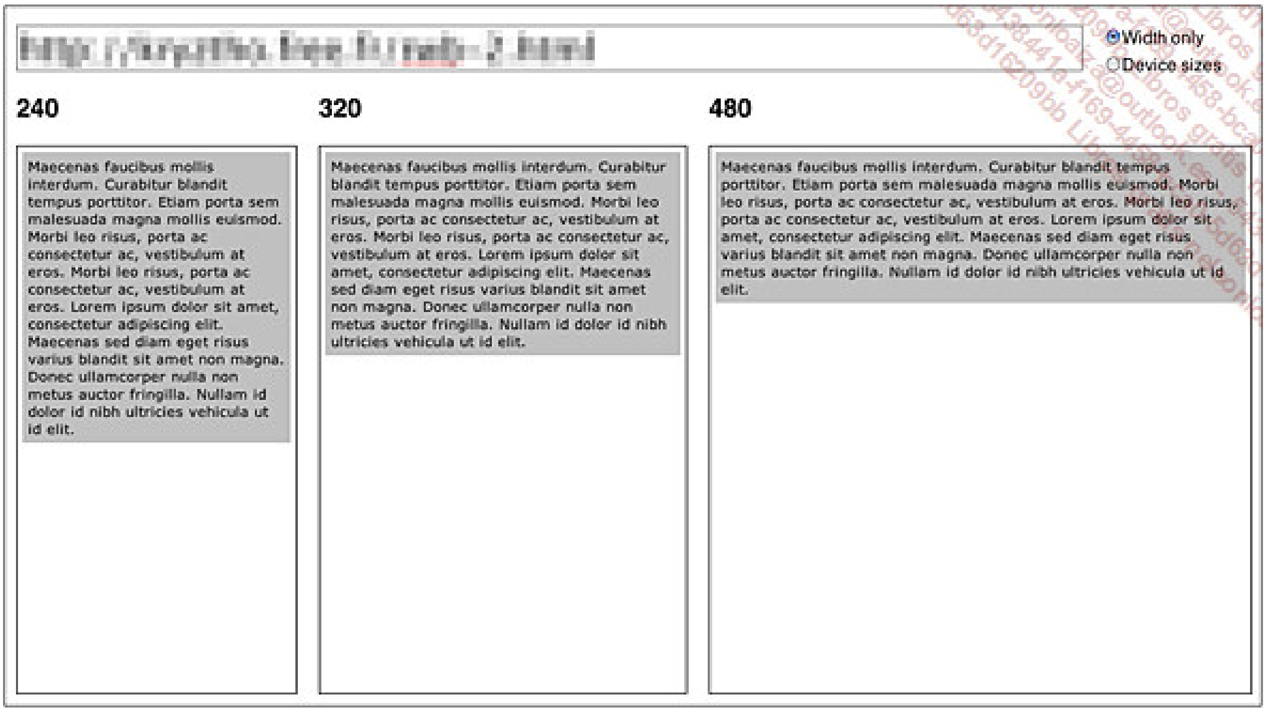
También puede usar aplicaciones en línea. El sitio web **Mobile Phone Emulator**(http://www.mobilephoneemulator.com/), como su nombre indica, permite examinar la visualización de una página web con diferentes dispositivos.



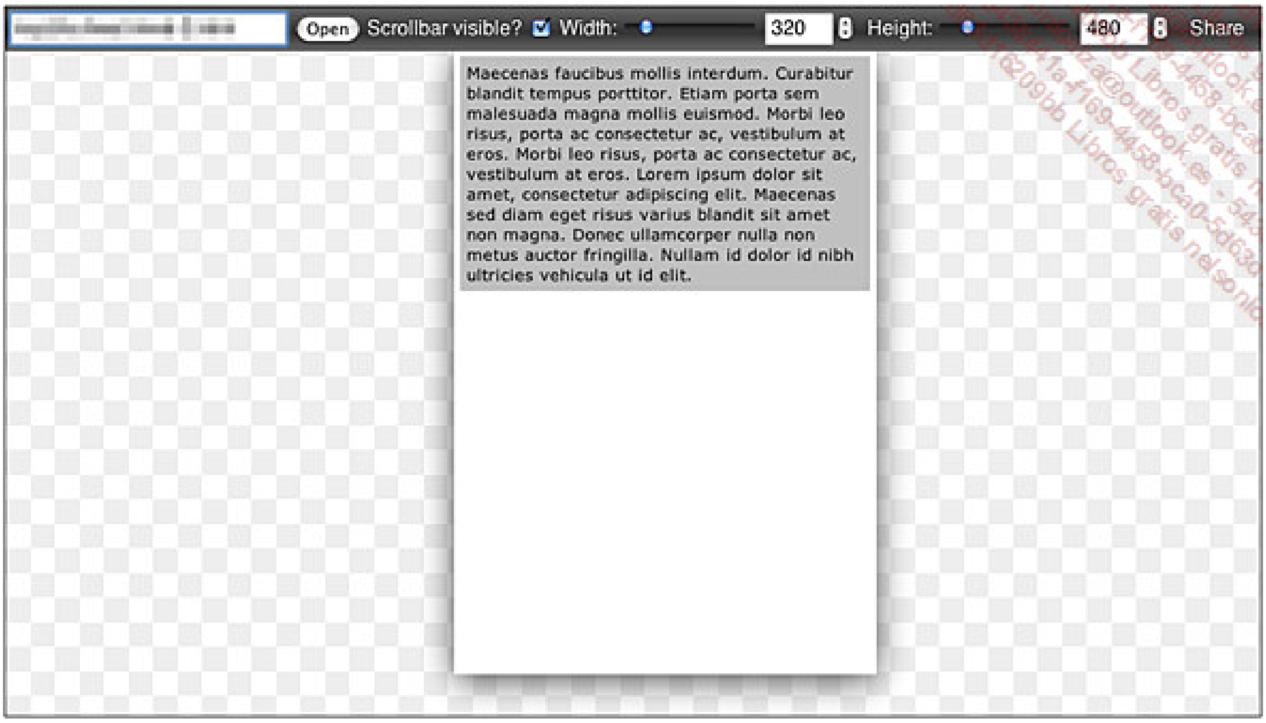
Otro emulador en línea es **Screenfly** (http://quirktools.com/screenfly/). En la página de inicio deberá indicar la URL que quiera examinar y, a continuación, con los botones-menú de la parte de abajo, deberá seleccionar el tipo de dispositivo que quiera simular.



El sitio web **Responsive Design Testing** (http://mattkersley.com/responsive/) le permite visualizar una URL con diferentes tamaños preestablecidos.



El sitio web **Responsivepx** (http://responsivepx.com/) le permite difinir usted mismo el ancho y el alto de pantalla que quiera simular.



Por último, en la URL http://mobiforge.com/testing/story/a-guide-mobile-emulators encontrará una larga lista de herramientas para emular todo tipo de smartphones.

Las estructuras flexibles

1. Las técnicas

Los sitios web que no se preocup(ab)an en tener un diseño adaptable presentan, en su gran mayoría, una arquitectura basada en dimensiones fijas. Una u otra caja `<div>` presentará entre sus propiedades CSS un ancho establecido con un valor fijo: `width: 730px`. Esto hace que el tamaño de la caja siempre sea el mismo, independientemente del tamaño de la pantalla en la cual se visualice. Así, si se accede al sitio web a partir de un smartphone o de una tableta, las dimensiones no variarán, puesto que son fijas, y el sitio web no se adaptará en absoluto.

Para que el diseño de un sitio web sea adaptable, será necesario que las dimensiones de los contenedores (`<div>`, `<nav>`, `<header>`...) estén indicadas en unidades relativas: en porcentajes.

Tendremos entonces una sintaxis de tipo:

```
header#inicio {
    ...
    width: 75%;
    ...
}
```

Esto quiere decir que el elemento `<header>`, que tiene el código de identificación `#inicio`, se visualizará con un ancho del 75% del ancho de su elemento padre.

Eso implica que usted, el diseñador web, deberá corregir todos los elementos estructurales (`<div>`, `<nav>`, `<header>`...) que tengan dimensiones fijas, para asignarles dimensiones relativas en función de una búsqueda Media Queries.

Además, puede usar la palabra clave `!important`, en sus reglas CSS modificadas, para que tengan prioridad frente a cualquier otra regla declarada de manera "habitual", sin búsqueda de medios de difusión.

También puede ser interesante fijar un ancho máximo para los elementos de contenido (imágenes, tablas...) con la propiedad `max-width: 100%`, para que su tamaño nunca sea mayor que el de su elemento padre.

Igualmente, deberá prestar atención al texto que sea más largo que su contenedor. Utilice para ello la propiedad `word-wrap: break-word`.

Por último, use siempre tamaños relativos para los caracteres con las unidades `em` o `%`.

2. Un ejemplo sencillo

Vamos a crear una caja `<div>` contenedor (`#contenedor`) que va a contener otras dos cajas`<div>` (`#izquierda` y `#derecha`). El contenedor tendrá un tamaño flexible del 75% respecto a su elemento padre, el `<body>`. La caja `#izquierda` tendrá un ancho flexible igual al 60% de su elemento padre `#contenedor`. La caja `#derecha` tendrá un ancho flexible igual al 40% de su elemento padre `#contenedor`.

Evidentemente, la suma de los dos porcentajes no deberá ser superior a 100%.

Esta es la parte HTML:

```
<div id="contenedor">
    <div id="izquierda">
        <p>Feugait qui...</p>
    </div>
```

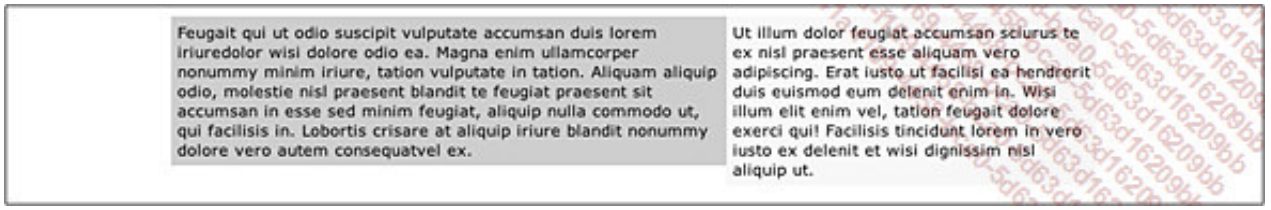
```
    <div id="derecha">
        <p>Ut illum dolor...</p>
    </div>
</div>
```

Esta es la parte CSS:

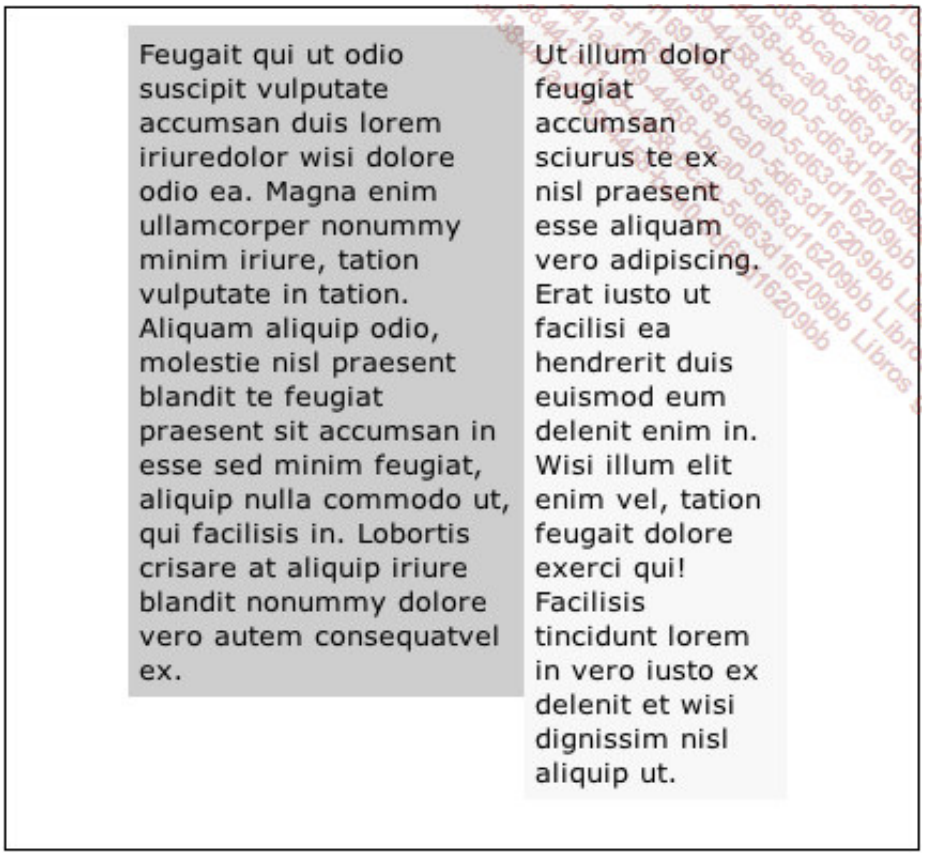
```
body {
    font: .8em Verdana, Arial, Helvetica, sans-serif;
}
p {
    margin: 5px;
}
#contenedor{
    width: 75%;
    margin: 0 auto;
}
#izquierda {
    float: left;
    width: 60%;
    background-color: #cecece;
}
#derecha{
    float: left;
    width: 40%;
    background-color: #f8f8f8;
}
```

El contenedor (`#contenedor`) y las dos cajas que contiene (`#izquierda` y `#derecha`) tendrán siempre un tamaño flexible, proporcional al ancho del `<body>`, en función del tamaño de la ventana del navegador.

Así se visualizará con una pantalla de gran tamaño:



Así se visualizará con una pantalla de menor tamaño:



3. Ejemplo de un sitio web sencillo

Vamos a crear un sitio web con un diseño flexible. Será un sitio web muy sencillo, con un encabezado (`<header>`), un menú de navegación (`<nav>`), una columna (`<section>`), en la parte izquierda, con tres artículos (`<article>`), una barra lateral (`<aside>`), en la parte derecha, con una imagen y una lista. Y, por último, un pie de página (`<footer>`).

Este sería el código HTML5 de la página web:

```
<!DOCTYPE HTML>
<html lang="es">
<head>
<title>Título de la página web</title>
<meta charset="UTF-8" />
<link rel="stylesheet" href="rwd-1.css" />
</head>
<body>
<div id="contenedor">
     <header id="arriba">
         <h1>Mi sitio web</h1>
         <h2>El eslogan de mi sitio web</h2>
     </header>
     <nav id="menu-navegacion">
     <p><a href="#">Vínculo 1</a> | <a href="#">Vínculo 2</a> |
<a href="#">Vínculo 3</a> | <a href="#">Vínculo 4</a> |
<a href="#">Vínculo 5</a> | <a href="#">Vínculo 6</a></p>
     </nav>
     <section id="contenido">
         <article>
             <h1>Mi primer artículo</h1>
             <p>Morbi leo risus...</p>
             <p>Integer posuer...</p>
         </article>
         <article>
```

```
            <h1>Mi segundo artículo</h1>
            <p>Etiam porta...</p>
            <p>Lorem ipsum...</p>
        </article>
        <article>
            <h1>Mi tercer artículo</h1>
            <p>Morbi leo...</p>
            <p>Integer posuere...</p>
        </article>
    </section>
    <aside id="sidebar">
        <p><img src="tigre.jpg" alt="Un tigre" /></p>
        <ul>
           <li>Primero</li>
           <li>Segundo</li>
           <li>Tercero</li>
           <li>Cuarto</li>
        </ul>
    </aside>
    <footer id="piedepagina">
           <p>Concepción y realización: yo mismo. Contacto:
<a href="mailto:contacto@midominio.org">contacto@midominio.org</a></p>
    </footer>
</div>
</body>
</html>
```

Estos serían los estilos CSS3:

```
/* Estilo general */
body {
    padding: 0;
    background-color: gray;
    font: .8em Verdana, Arial, Helvetica, sans-serif;
}
body, h1, h2, p {
    margin: 0;
}
div#contenedor{
    width: 940px;
    margin: 20px auto 0 auto;
    padding: 10px;
    background-color: white;
    border-radius: 10px;
}
/* El encabezado */
header#arriba {
    margin: 0 0 10px 0;
    padding: 5px;
    background-color: gray;
    border-radius: 10px;
}
header#arriba h1, header#arriba h2 {
    color: white;
}
/* El menú de navegación */
nav#menu-navegacion {
    margin: 0 0 10px 0;
    padding: 5px;
    background-color: silver;
    border-radius: 10px;
}
nav#menu-navegacion p {
    color: white;
}
```

```
nav#menu-navegacion a {
     text-decoration: none;
     color: white;
}
/* El contenido principal */
section#contenido{
     float: left;
     width: 720px;
     padding: 5px;
}
article {
     margin: 0 0 20px 0;
     border-top: 1px dotted #333;
}
/* La columna lateral */
aside#sidebar {
     float: left;
     width: 190px;
     margin: 0 0 0 10px;
     padding: 5px;
     border-radius: 5px;
     background-color: #E8E8E8;
}
aside#sidebar p {
     text-align: center;
}
/* El pie de página */
footer#piedepagina {
     padding: 5px;
     border-radius: 5px;
     background-color: #E8E8E8;
     clear: both;
}
footer#piedepagina p {
     text-align: center;
     font-size: .75em;
}
```

Este sería el resultado en una pantalla de ordenador:

Este sería el resultado en un iPhone:

4. Aplicar un diseño adaptable sencillo

En un primer momento, tendremos que añadir la línea para el `viewport` en la cabecera de la página **HTML**, en el `<head>`.

```
<head>
    ...
<meta name="viewport" content="width=device-width" />
    ...
</head>
```

En el archivo **CSS**, vamos a insertar una búsqueda Media Queries, al final de la página. Vamos a seleccionar la difusión en pantalla para un tamaño máximo.

```
@media screen and (max-width: 768px) {
   ...
}
```

¿Por qué he elegido el tamaño máximo de 768 píxeles? Simplemente, porque ese es el ancho de un iPad.

Queremos que el fondo de la página sea blanco, y no gris, como en la versión para la pantalla de ordenador.

```
@media screen and (max-width: 768px) {
     body {
         background-color: #fff;
```

```
    }
}
```

Queremos que la caja , en lugar de tener un tamaño fijo, tenga un tamaño relativo, para que ocupe todo el espacio disponible en su elemento padre <body>.

```
@media screen and (max-width: 768px) {
    body {
        background-color: #fff;
    }
    div#contenedor{
        width: 100%;
        margin: 0;
    }
}
```

El color de fondo del banner de la parte de arriba será negro, con el texto en blanco.

```
@media screen and (max-width: 768px) {
    body {
        background-color: #fff;
    }
    div#contenedor{
        width: 100%;
        margin: 0;
    }
    header#arriba{
        background-color: #000;
        color: #fff;
    }
}
```

Los dos contenedores (<section> y <aside>) ya no podrán ser flotantes, para ganar espacio y tener así contenedores que se superpongan verticalmente.

```
@media screen and (max-width: 768px) {
    body {
        background-color: #fff;
    }
    div#contenedor {
        width: 100%;
        margin: 0;
    }
    header#arriba{
        background-color: #000;
        color: #fff;
    }
    section#contenido{
        float: none;
        width: auto;
    }
    aside#sidebar {
        float: none;
        width: auto;
        margin: 0 0 10px 0;
    }
}
```

No queremos mostrar la imagen (<img>) en la barra lateral, una vez más, para ganar espacio y concentrarnos en el contenido.

```
@media screen and (max-width: 768px) {
    body {
```

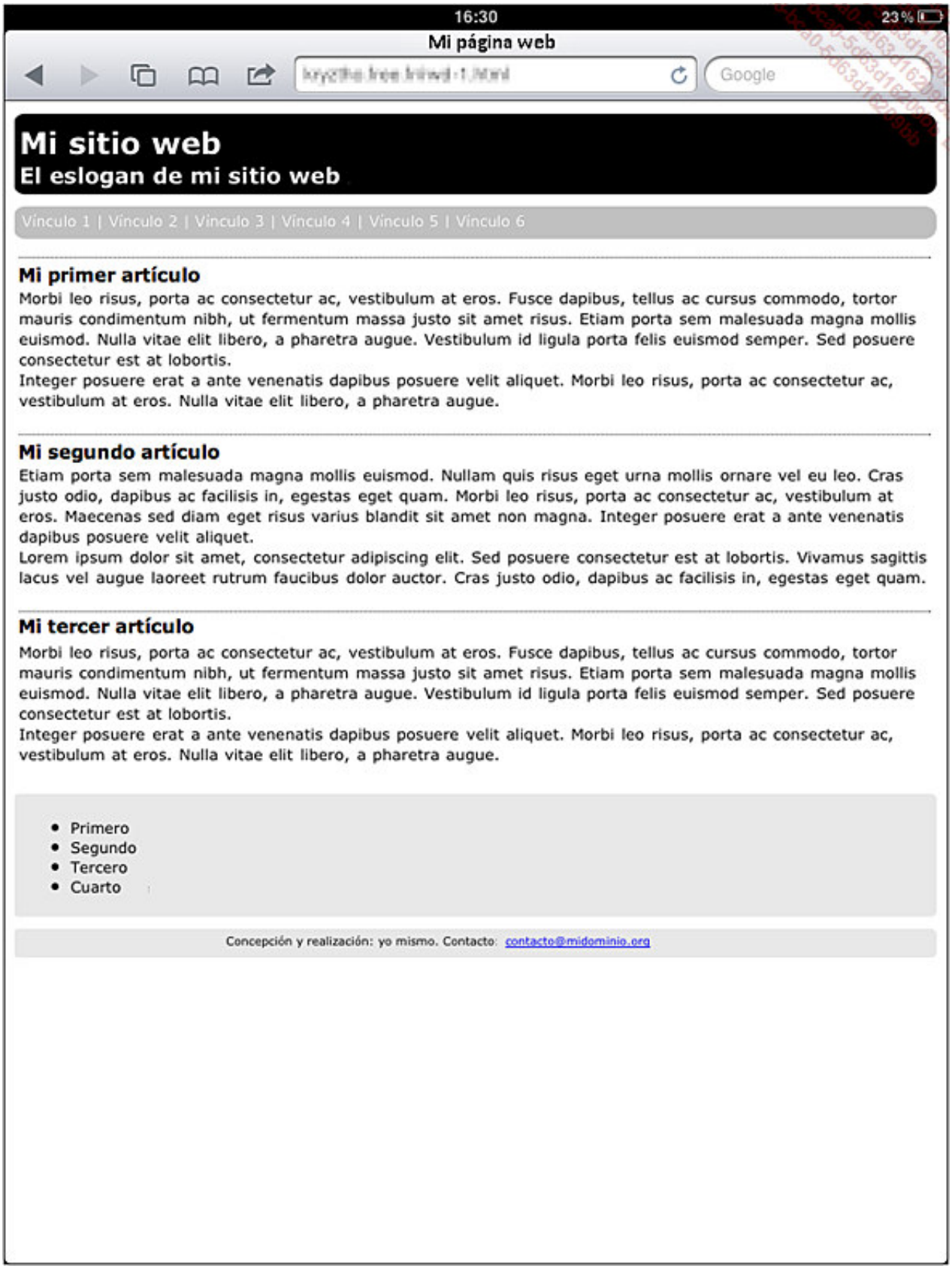
```
        background-color: #fff;
    }
    div#contenedor{
        width: 100%;
        margin: 0;
    }
    header#arriba{
        background-color: #000;
        color: #fff;
    }
    section#contenido{
        float: none;
        width: auto;
    }
    aside#sidebar {
        float: none;
        width: auto;
        margin: 0 0 10px 0;
    }
    aside#sidebar img {
        display: none;
    }
}
```

Por último, vamos a adaptar el tamaño de los títulos <h1> de los artículos.

```
@media screen and (max-width: 768px) {
    body {
        background-color: #fff;
    }
    div#contenedor{
        width: 100%;
        margin: 0;
    }
    header#arriba {
        background-color: #000;
        color: #fff;
    }
    section#contenido {
        float: none;
        width: auto;
    }
    aside#sidebar {
        float: none;
        width: auto;
        margin: 0 0 10px 0;
    }
    aside#sidebar img {
        display: none;
    }
    article h1 {
        font-size: 1.5em;
    }
}
```

Así se visualizará en un iPhone 4:

Así se visualizará en un iPad:



5. Las plantillas flexibles

Estas son algunas URL que tratan sobre plantillas adaptables:

- **Skeleton** (http://getskeleton.com/): *A Beautiful Boilerplate for Responsive, Mobile-Friendly Development*.
- **Columnal** (http://www.columnal.com/): *A responsive CSS grid system helping desktop and mobile browsers play nicely together*.

- **Significantpixels** (http://www.significantpixels.com/2010/12/10/responsive-grid-systems/):*Responsive Grid Systems - A Conceptual Framework for Extending Upon the Functionality of 960.gs*.
- **Golden Grid System** (http://goldengridsystem.com/): *A folding grid for responsive design*.

Las imágenes flexibles

1. El desbordamiento de las imágenes

Las imágenes también pueden presentar problemas de dimensión respecto a su elemento padre. Si el tamaño de la imagen es mayor que el de su contenedor, esta "se saldrá" de su elemento padre.

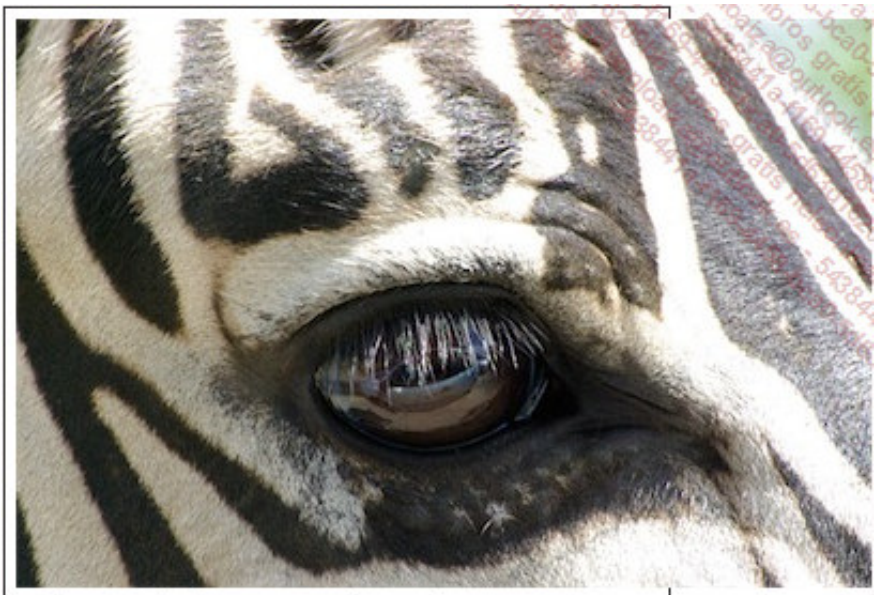
Veamos un ejemplo simple de una caja `<div id="miCaja">`, que contiene texto y una imagen.

```
<div id="miCaja">
    <p><img src="cebra.jpg" alt="Una cebra" /><br />Delenit odio
luptatum illum dignissim...</p>
</div>
```

Y un estilo básico:

```
body {
    font: .8em Verdana, Arial, Helvetica, sans-serif;
}
#miCaja {
    float: left;
    width: 300px;
    border: 1px solid #333;
    padding: 5px;
}
p {
    margin: 0;
}
```

Este es el resultado obtenido: la fotografía de la cebra se sale de la caja, su tamaño es mayor que el de su elemento padre.



Delenit odio luptatum illum dignissim ullamcorper in aliquip, dolor delenit dolore? Ad, ex wisi sciurus elit in. Facilisis hendrerit esse ex aliquip iriuredolor nibh aliquip exerci, aliquip dolore eum. Commodo illum, quis tincidunt et minim sed feugiat amet commodo at et ullamcorper erat, commodo illum vulputate elit in. Crisare, te nulla autem in sed lorem et vel dolore in, eros diam dolor. Quis esse tation adipiscing qui lobortis et molestie dignissim tation ullamcorper blandit veniam eros autem, iriuredolor ullamcorper consectetuer.

2. Imponer un ancho máximo

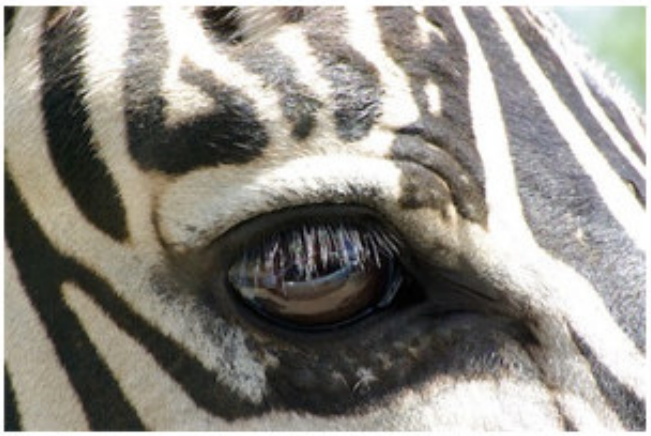
La solución para evitar ese problema, y el de la redimensión generada por las búsquedas Media Queries, consiste en imponer un ancho máximo en relación al elemento contenedor, el elemento padre de la imagen. Simplemente tendrá que usar la propiedad `max-width`.

Apliquemos ahora este estilo:

```
#miCaja img {
    max-width: 100%;
}
```

Estamos pidiendo que el tamaño máximo de las imágenes insertadas en la caja `<div id="miCaja">` sea del 100%. De este modo, las imágenes nunca tendrán un tamaño mayor al de su elemento padre.

Y el resultado obtenido:



Delenit odio luptatum illum dignissim ullamcorper in aliquip, dolor delenit dolore? Ad, ex wisi sciurus elit in. Facilisis hendrerit esse ex aliquip iriuredolor nibh aliquip exerci, aliquip dolore eum. Commodo illum, quis tincidunt et minim sed feugiat amet commodo at et ullamcorper erat, commodo illum vulputate elit in. Crisare, te nulla autem in sed lorem et vel dolore in, eros diam dolor. Quis esse tation adipiscing qui lobortis et molestie dignissim tation ullamcorper blandit veniam eros autem, iriuredolor ullamcorper consectetuer.

Claro está, podrá elegir el valor que usted prefiera. Este es un ejemplo con un valor del 50 %:

```
#miCaja img {
	max-width: 50%;
}
```

Y el resultado visual:

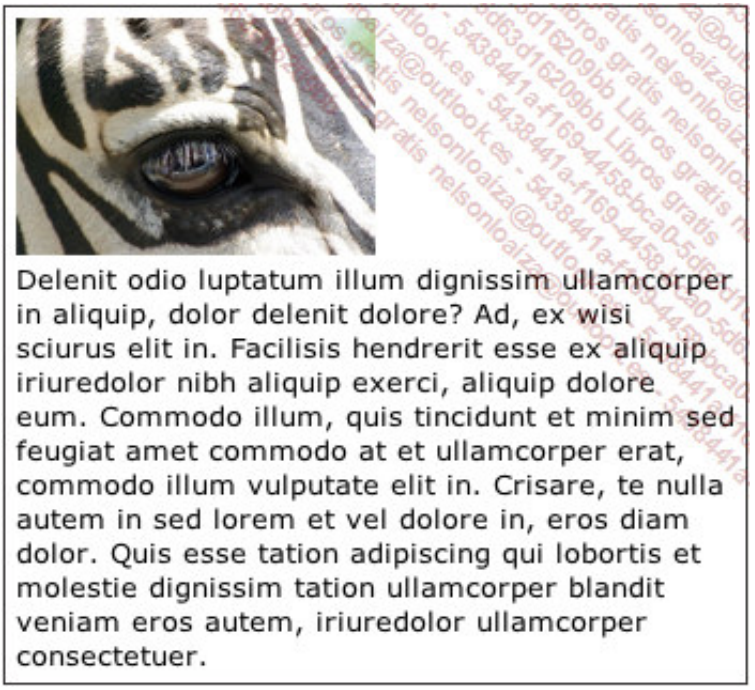
Delenit odio luptatum illum dignissim ullamcorper in aliquip, dolor delenit dolore? Ad, ex wisi sciurus elit in. Facilisis hendrerit esse ex aliquip iriuredolor nibh aliquip exerci, aliquip dolore eum. Commodo illum, quis tincidunt et minim sed feugiat amet commodo at et ullamcorper erat, commodo illum vulputate elit in. Crisare, te nulla autem in sed lorem et vel dolore in, eros diam dolor. Quis esse tation adipiscing qui lobortis et molestie dignissim tation ullamcorper blandit veniam eros autem, iriuredolor ullamcorper consectetuer.

También podemos establecer que la caja tenga un tamaño flexible. En el siguiente ejemplo vamos a definir un ancho del 25%, en relación al `<body>`.

```
#miCaja {
    float: left;
    width: 25%;
    border: 1px solid #333;
    padding: 5px;
}
```

La caja y la imagen tienen ahora valores flexibles y su tamaño de visualización será proporcional (respectivamente del 25% y del 50%) respecto al tamaño del <body>, es decir, al tamaño de la ventana del navegador.

Así se visualizará en una pantalla de gran tamaño:



Así se visualizará en una pantalla de menor tamaño:

Delenit odio luptatum illum dignissim ullamcorper in aliquip, dolor delenit dolore? Ad, ex wisi sciurus elit in. Facilisis hendrerit esse ex aliquip iriuredolor nibh aliquip exerci, aliquip dolore eum. Commodo illum, quis tincidunt et minim sed feugiat amet commodo at et ullamcorper erat, commodo illum vulputate elit in. Crisare, te nulla autem in sed lorem et vel dolore in, eros diam dolor. Quis esse tation adipiscing qui lobortis et molestie dignissim tation ullamcorper blandit veniam eros autem, iriuredolor ullamcorper consectetuer.

Crear un diseño flexible

1. Las dos estrategias

Usted tiene un proyecto de sitio web y desea que los usuarios puedan consultar su contenido desde cualquier tipo de dispositivo: pantallas de ordenador, tabletas o smartphones.

Dispone de dos estrategias:

- Crear páginas web específicas para cada tipo de dispositivo.
- Realizar un diseño web adaptable.

Los adeptos de la primera estrategia afirman que el contexto de uso hace necesaria la creación de páginas web específicas. Según ellos, una persona que consulte un sitio web en una pantalla de ordenador deberá poder acceder a toda la información disponible, sin restricciones, ya que el contexto de uso, en la pantalla del ordenador, permite visualizarla completamente. Al contrario, si un usuario consulta un sitio web a partir de un smartphone, el contexto de uso no le permitirá visualizar toda la información disponible y el simple hecho de que esa persona utilice un smartphone sugiere que no necesita acceder a todo el contenido. Existen otros muchos argumentos para no adoptar el diseño web adaptable. Le recomiendo que lea este largo y bien argumentado artículo con el título "*11 reasons why Responsive Design isn't that cool!*", que encontrará en esta URL:http://www.webdesignshock.com/responsive-design-problems/

Los adeptos de la segunda solución afirman, por su parte, que no es posible saber por adelantado, en función del contexto de uso (ordenador, tableta o smartphone), a qué quieren tener acceso los visitantes de un sitio web. Por ese motivo, no podemos "censurar" el sitio web y solamente publicar una parte de su contenido. Además, ¿qué contenido se deberá mostrar y cuál no? Cada visitante es un mundo y las necesidades pueden ser muy diferentes. Si tenemos que ocultar determinada información, siempre podremos usar la propiedad `display: none`.

2. Diseñar un sitio web adaptable

Para diseñar un sitio web adaptable, deberá hacerse algunas preguntas:

- ¿Qué diferencias habrá entre el diseño de la versión para ordenador y la versión para smartphone del sitio web?
- ¿Dónde deberán posicionarse las principales zonas de visualización?
- ¿Qué contenido deberá mostrarse imperativamente en la versión para smartphone y qué contenido no se mostrará?
- La misma pregunta para los elementos multimedia: imágenes, audio, vídeos, PDF...
- ¿Cómo se van a mostrar los elementos de navegación?
- ¿Cómo se van a mostrar los formularios?
- ¿Esos cambios van a afectar a la ergonomía?
- ...

A continuación podrá comenzar a crear las plantillas para los diferentes soportes, para ver el resultado y modificarlo, adaptarlo, en función de las necesidades específicas. Lo ideal es usar las plantillas directamente en los diferentes soportes para probar la navegación, la ergonomía y la usabilidad del sitio web que esté creando.

Es primordial que ataque por "todos los frentes" al mismo tiempo, es decir, que trabaje en todas las plantillas al mismo tiempo: la plantilla para las pantallas de ordenador, la de las tabletas y la de los smartphones. Sería un error lamentable dedicarse exclusivamente a la maqueta "pantalla de ordenador" y luego intentar adaptarla a las demás pantallas. Eso no funcionará. Es necesario tener

una visión global del proyecto.

3. Otros ejemplos de diseño web adaptable

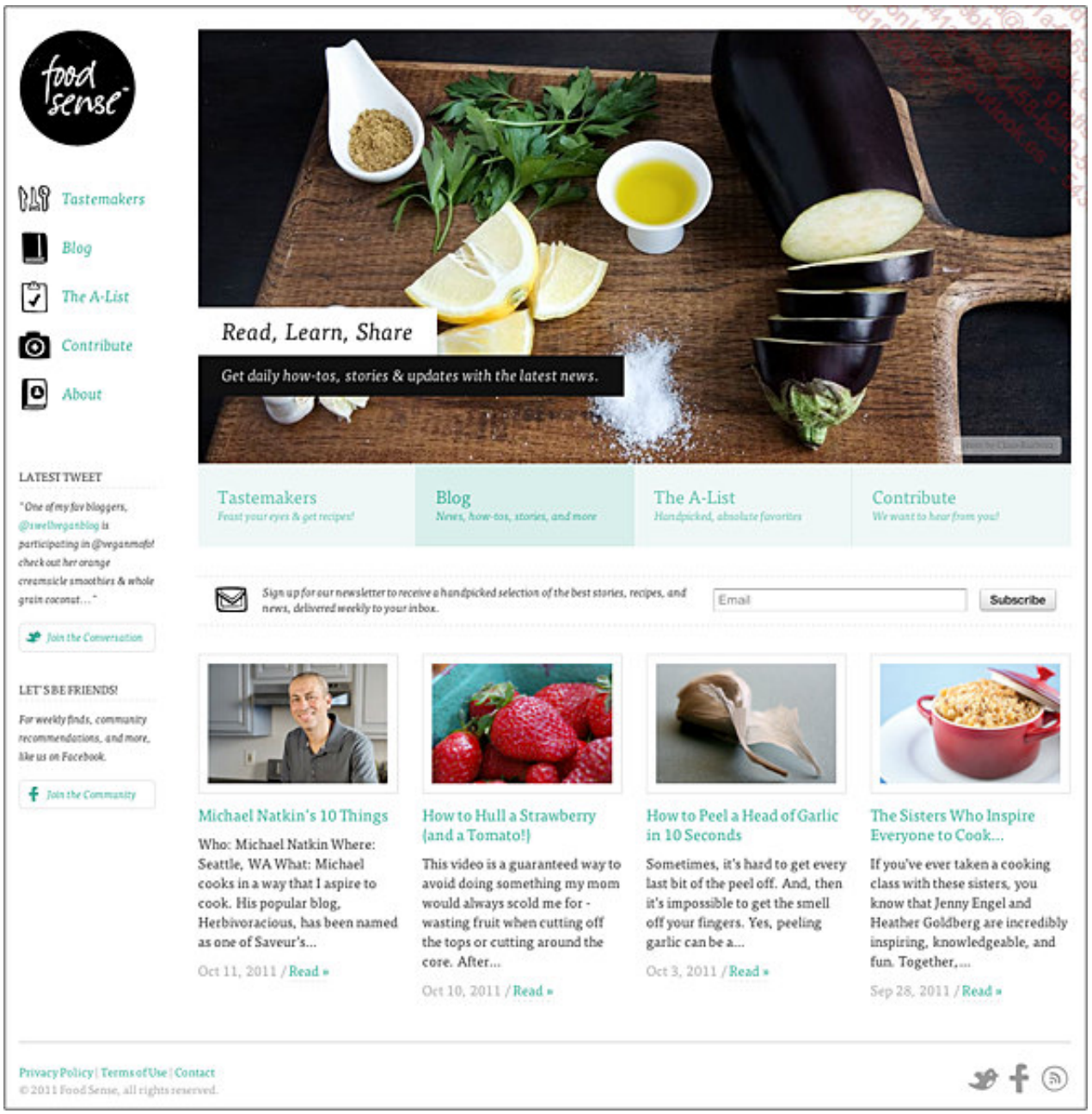
En la web encontrará multitud de sitios web que recopilan aquellos sitios web que utilizan las búsquedas Media Queries. Uno de los más famosos es **Media Queries - a collection of sites using media queries**, con su magnífica URL, disfrútelo: http://mediaqueri.es/

El sitio web Food Sense

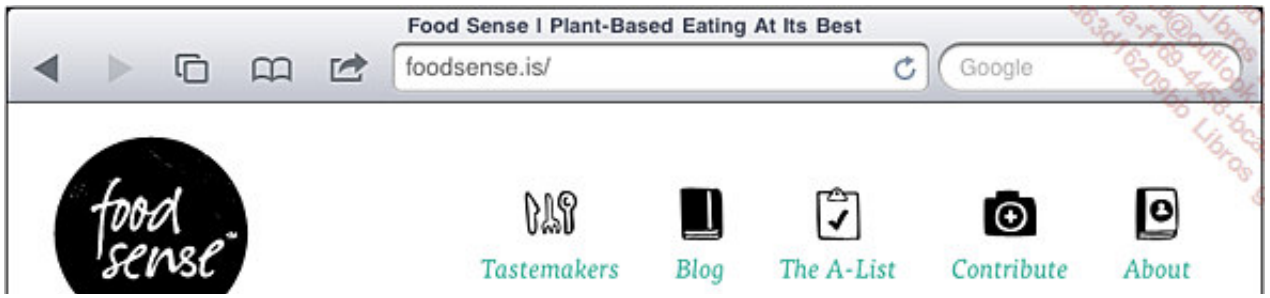
1. La visualización

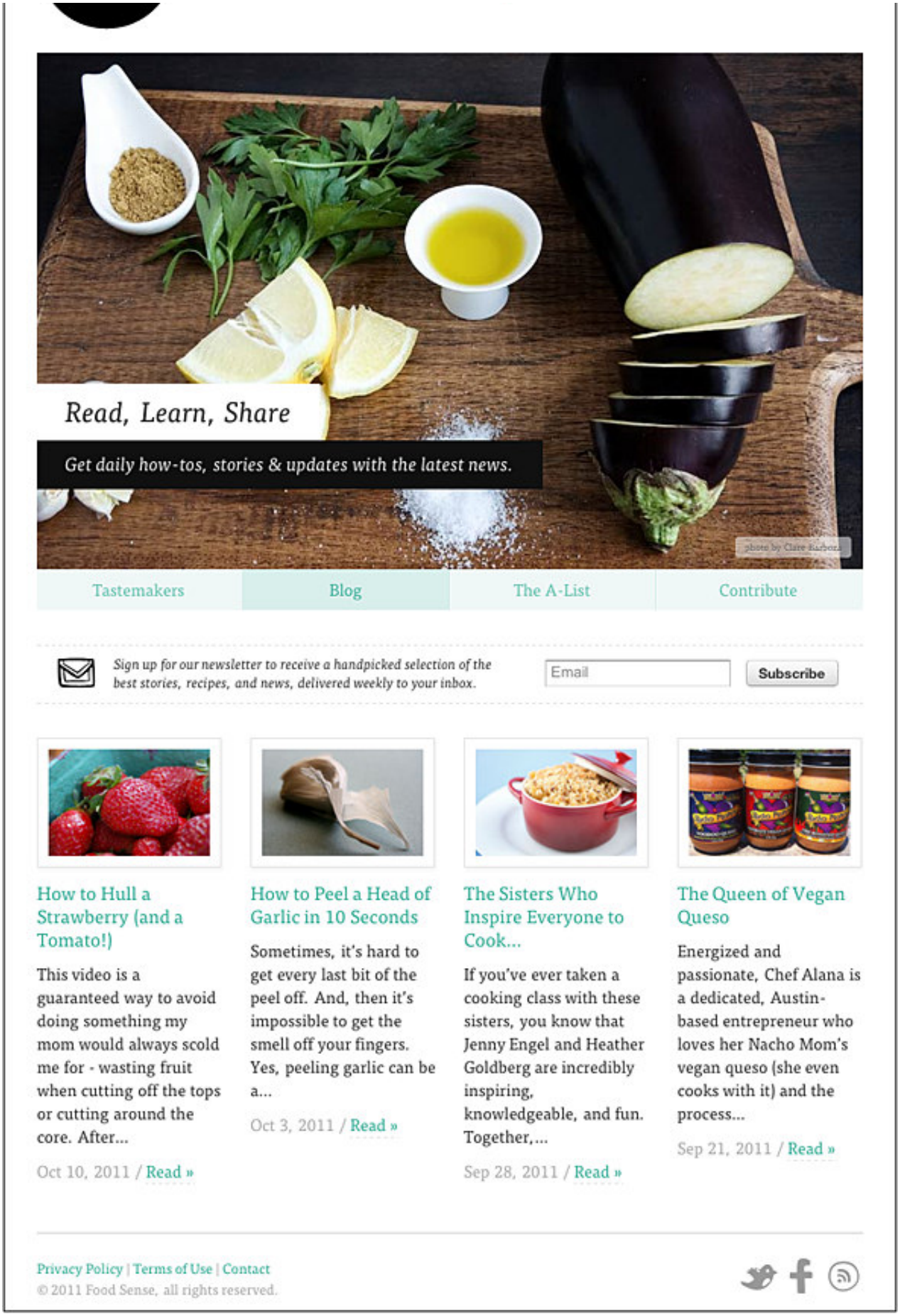
Este sitio web (http://foodsense.is/) utiliza las búsquedas de medios de difusión y se adapta para optimizar su visualización en función del dispositivo utilizado.

Así se visualizará en una pantalla de ordenador.

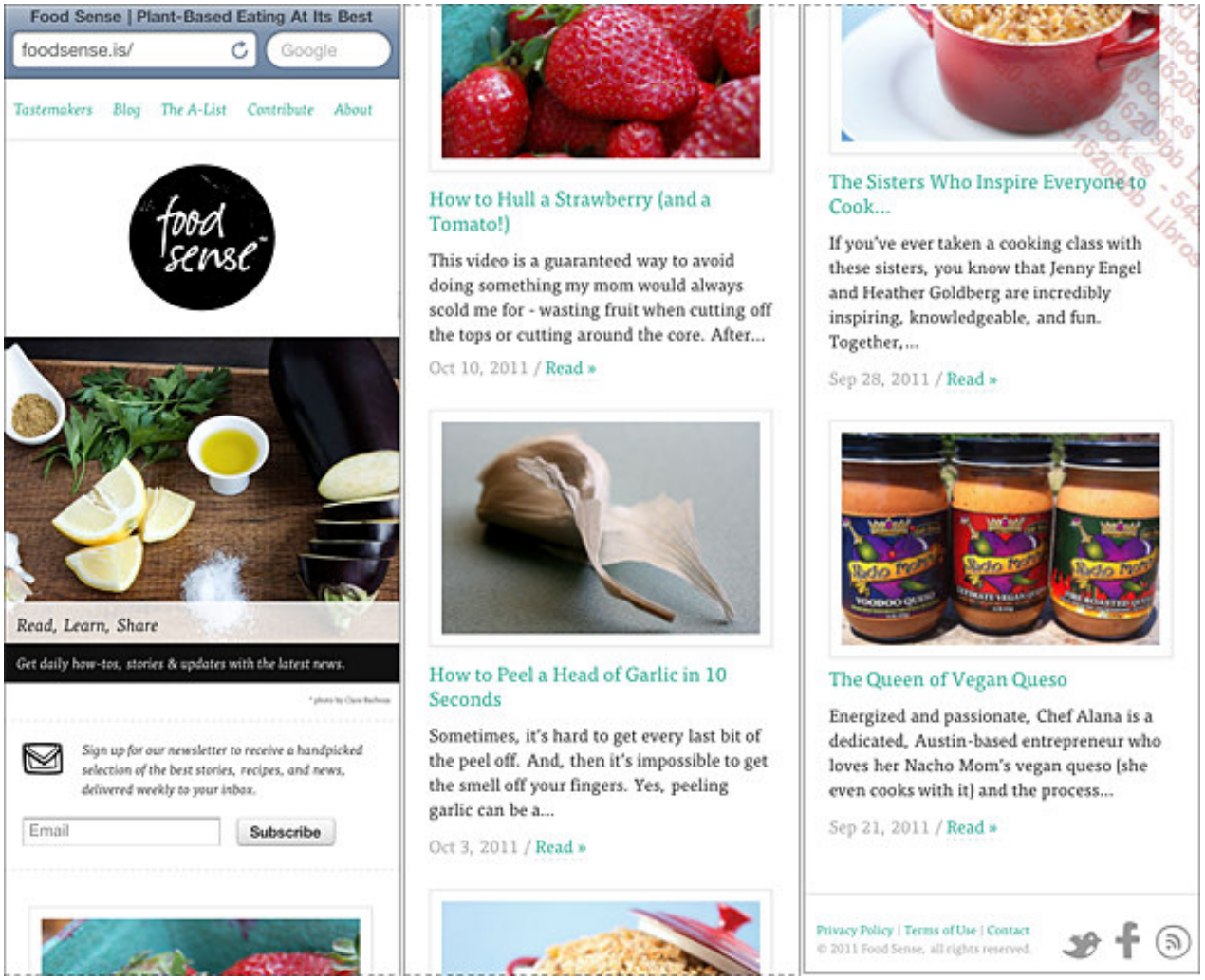


Así se visualizará en un iPad:





Así se visualizará en un iPhone:



2. La estructura principal

Esta es la estructura principal del sitio web en HTML5:

Estos son los principales estilos CSS que se han utilizado:

El contenedor general es una caja `<div id="wrap">`:

```
#wrap {
    position: relative;
    width: 100%;
    height: 100%;
    margin: 0 auto;
    background: #FFF;
}
```

El encabezado es un `<header>`, que contiene un `<h1>` con una imagen de fondo:

```
header[role="masthead"] {
    float: left;
    width: 160px;
    margin-bottom: 25px;
}
h1#branding {
    display: block;
    width: 120px;
    height: 120px;
```

```
    text-indent: -999em;
    background: url(../img/food-sense.png) no-repeat 0 0;
    -webkit-transition: all .3s ease;
    -moz-transition: all .3s ease;
    -o-transition: all .3s ease;
    transition: all .3s ease;
}
```

Los vínculos de navegación de la parte izquierda se encuentran dentro de un elemento <nav>:

```
nav {
    position: absolute;
    top: 158px;
    left: 0;
    width: 160px;
}
```

Los elementos para las redes sociales se encuentran dentro de un elemento <aside>, dentro de los elementos <section>:

```
aside[role="social"] {
    position: absolute;
    top: 452px;
    left: 0;
    width: 160px;
}
aside[role="social"] section {
    display: block;
    margin-bottom: 32px;
    padding-right: 20px;
}
```

La zona central es una caja :

```
#content {
    position: relative;
    float: right;
    width: 712px;
}
```

La zona de visualización del diaporama se encuentra dentro de un elemento <section id="featured">:

```
#featured {
    position: relative;
    width: 100%;
    height: 510px;
}
```

El diaporama se ha insertado en una caja :

```
#featured #slides {
    width: 712px;
    height: 446px;
    overflow: hidden;
}
```

El menú de navegación del diaporama es una caja :

```
#featured #nav {
    position: absolute;
    top: 445px;
```

```
    width: 712px;
    background: #EFF8F7;
    overflow: hidden;
    -webkit-border-bottom-left-radius: 3px;
    -moz-border-bottom-left-radius: 3px;
    border-bottom-left-radius: 3px;
    -webkit-border-bottom-right-radius: 3px;
    -moz-border-bottom-right-radius: 3px;
    border-bottom-right-radius: 3px;
}
```

La zona de inscripción a la newsletter se encuentra dentro de un elemento `<section id="newsletter">`:

```
#newsletter {
    overflow: hidden;
    clear: both;
    width: 100%;
    min-height: 48px;
    margin-bottom: 29px;
    line-height: 1em;
    border-top: 1px dashed #E5E5E5;
    border-bottom: 1px dashed #E5E5E5;
    background: url(../img/icon-mailinglist-32.png) no-repeat
18px 7px;
}
```

Los cuatro artículos están insertados en elementos `<article>`, dentro de un elemento`<section id="recent">`:

```
#recent {
    width: 100%;
    clear: both;
    margin-bottom: 18px;
    overflow: hidden;
}
#recent article {
    float: left;
    width: 160px;
    margin: 0 24px 25px 0;
}
```

Por último, el pie de página es un elemento `<footer>`:

```
footer[role="siteinfo"] {
    clear: both;
    position: relative;
    min-height: 70px;
    padding-top: 20px;
    border-top: 2px solid #E5E5E5;
    color: #AAA;
}
```

3. El viewport

El viewport es muy sencillo:

```
<meta name="viewport" content="width=device-width;
initial-scale=1.0">
```

4. La visualización para las pantallas de mayor tamaño

El diseñador ha utilizado Media Queries para las pantallas muy grandes, de un ancho de 1176 píxeles como mínimo:

```
/* WIDE DESKTOP LAYOUT
----------------------------------------- */
@media only screen and (min-width: 1176px) {
    ...
} /* CLOSE WIDE LAYOUT */
```

El diseño específico modifica, por ejemplo:

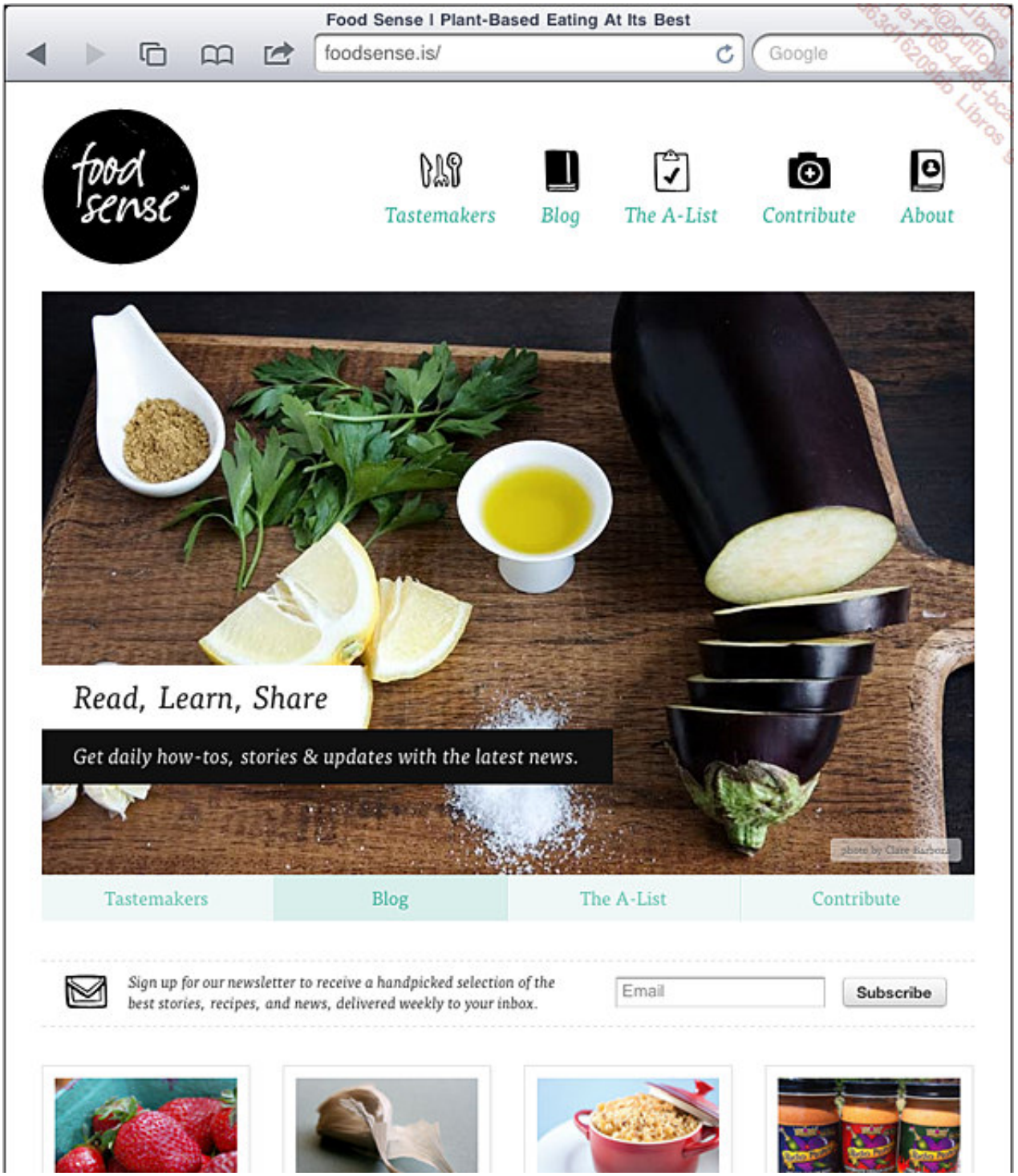
- El ancho fijo de `<body>`.
- El ancho del contenedor principal: `#content`.
- La altura de la zona central (diaporama y menú de navegación): `#featured`.
- El ancho de la caja `<div>` del diaporama: `#slides`.
- El ancho de la caja `<div>` del menú de navegación que está debajo del diaporama: `#nav`.
- El ancho de los artículos: `<article>`.

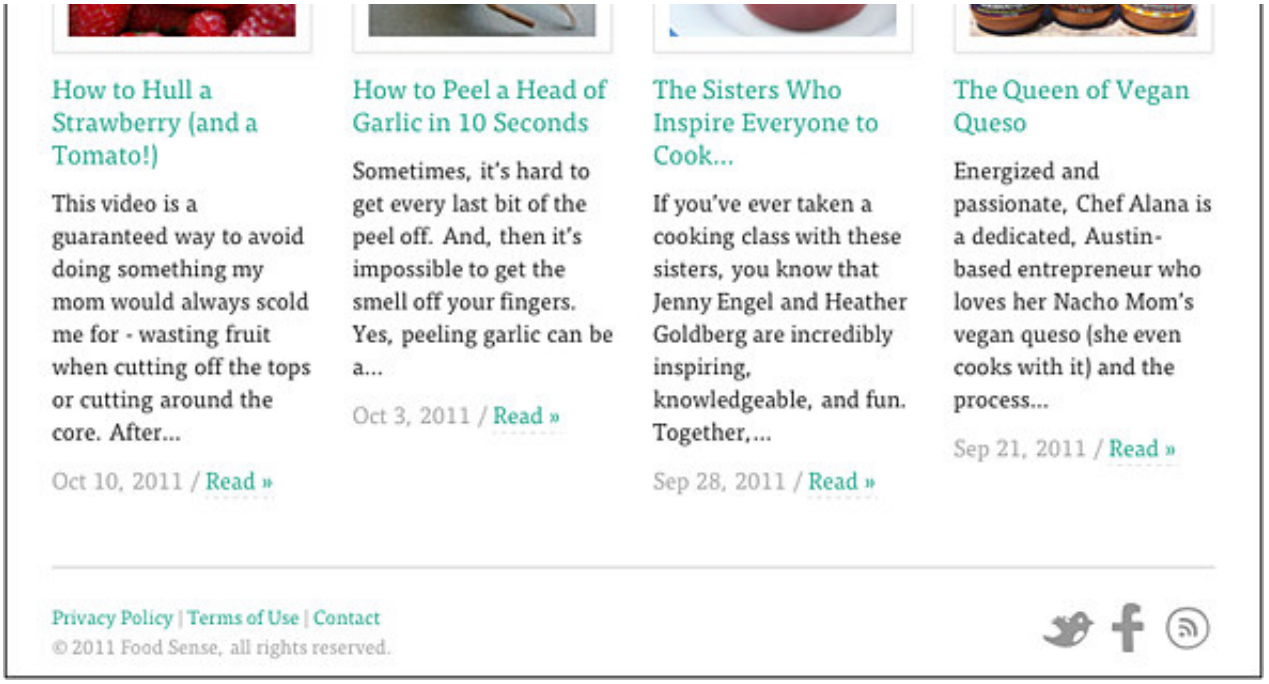
```
/* WIDE DESKTOP LAYOUT
----------------------------------------------- */
@media only screen and (min-width: 1176px) {
body { width: 1080px; }

#content {
    float: right;
    width: 896px;
}
/* FEATURED */
#featured { height: 560px; }
#featured #slides { width: 896px; }
#featured section img { left: 0; }

#featured section.alt h1,
#featured section.alt h2 {
    right: 0;
}
#featured section .credit { right: 10px; }

/* FEATURED NAV */
#featured #nav {
    top: 445px;
    width: 896px;
}
#featured #nav a {
    width: 191px;
    height: 60px;
    padding: 25px 12px 0 20px;
}
#featured #nav h3 { font-size: 1.125em; }
#featured #nav li { text-align: left; }
#featured #nav li p { display: inline; }

/* NEWSLETTER */
#newsletter input#email { width: 260px; }
/* RECENT */
#recent article { width: 206px; }
#recent article img { width: 185px; }
/* RECIPE GALLERY */
```

```
#content.gallery article {
    width: 184px;
    height: 430px;
}
#content.gallery article:nth-child(2n+2) { margin-right: 24px; }
#content.gallery #main article:nth-child(4n+4) { margin-right: 0; }
/* MISC */
.vendors article aside { width: 60%; }
#content aside input[type="search"] { width: auto; }
} /* CLOSE WIDE LAYOUT */
```

5. La visualización para las tabletas táctiles

Las modificaciones más destacadas son el nuevo posicionamiento del encabezado `<header>`, en lo alto de la ventana, al igual que para el menú de navegación, y la "desaparición" de la zona para las redes sociales `<aside>`.





Tenemos, por supuesto, una búsqueda **Media Queries** para la difusión del sitio web en las tabletas táctiles. Se trata de las pantallas que tiene un tamaño comprendido entre 768 píxeles como mínimo y 991 píxeles como máximo.

```
/* TABLET LAYOUT
----------------------------------------------- */
@media only screen and (min-width: 768px) and (max-width: 991px) {
    ...
} /* CLOSE TABLET LAYOUT */
```

El diseño específico modifica, por ejemplo:

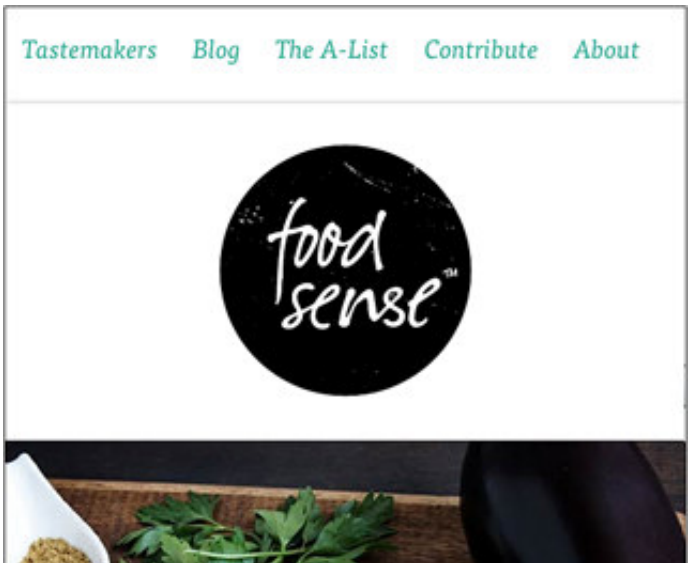
- El ancho fijo de `<body>`.
- El ancho del encabezado `<header>`, para su visualización en sentido horizontal.
- Un nuevo posicionamiento del menú de navegación, `<nav>`, que pasa así de su posición inicial a la izquierda, a posicionarse en la parte superior derecha, con una presentación horizontal.
- La altura de la zona central (diaporama y menú de navegación): `#featured`.
- El ancho de la caja `<div>` del diaporama: `#slides`.
- El ancho de la caja `<div>` del menú de navegación que se encuentra debajo del diaporama:`#nav`.
- La zona de navegación `<aside>`, que aparecía inicialmente en el lado izquierdo ha sido ocultada (`display: none`).

```
/* TABLET LAYOUT
----------------------------------------------- */
@media only screen and (min-width: 768px) and (max-width: 991px) {
body {
   width: 712px;
   padding: 20px 28px 0;
}
/* HEADER */
header[role="masthead"] {
   width: 712px;
   margin-bottom: 20px;
}
/* NAV */
```

```
nav {
   position: absolute;
   top: 70px;
   left: auto;
   right: 0;
   width: 528px;
   margin: 0;
}
nav ul { float: right; }
nav li { display: inline-block; }
nav a { display: inline; padding: 35px 15px 0 15px; }
nav li a { background-position: 50% 0; }
nav li a:hover { background-position: 50% 3px; }
/* FEATURED */
#featured { height: 510px; }
#featured #slides { width: 712px; }
#featured section img { left: -92px; }
#featured #nav {
    top: 445px;
    width: 712px;
}
#featured #nav a {
    width: 177px;
    height: 25px;
}
#featured #nav h3 {
    margin-bottom: 0;
    font-size: .875em;
}
#featured #nav li { text-align: center; }
#featured #nav li p { display: none; }
/* MISC */
#content header[role="page"] { padding-top: 0; }
aside[role="social"] { display: none; }
} /* CLOSE TABLET LAYOUT */
```

6. La visualización para los smartphones

La visualización para los smartphones es muy "clásica": todo el contenido se muestra en una sola columna. El menú de navegación solo muestra las etiquetas de los vínculos, sin los iconos. El logotipo del sitio web aparece debajo de este menú de navegación, centrado. La zona de visualización de las redes sociales, que inicialmente aparecía a la izquierda, se ha ocultado. Del mismo modo, el menú de navegación que inicialmente aparecía debajo del diaporama, ya no es visible.



Read, Learn, Share

Get daily how-tos, stories & updates with the latest news.

* photo by Clare Barboza

Sign up for our newsletter to receive a handpicked selection of the best stories, recipes, and news, delivered weekly to your inbox.

Email Subscribe

How to Hull a Strawberry (and a Tomato!)

This video is a guaranteed way to avoid doing something my mom would always scold [illegible] cutting off

The Queen of Vegan Queso

Energized and passionate, Chef Alana is a dedicated, Austin-based entrepreneur who loves her Nacho Mom's vegan queso (she even cooks with it) and the process...

Sep 21, 2011 / Read »

El diseñador web ha usado dos búsquedas específicas para los smartphones.

Una para la visualización en modo de retrato, con un ancho máximo de 767 píxeles.

```
/* MOBILE LAYOUT (PORTRAIT/320PX)
---------------------------------------------- */
@media only screen and (max-width: 767px) {
    ...
} /* CLOSE MOBILE LAYOUT */
```

Y otra para la visualización en modo de paisaje, con un ancho mínimo de 480 píxeles y un ancho máximo de 767 píxeles.

```
/* WIDE MOBILE LAYOUT (LANDSCAPE/480PX)
---------------------------------------------- */
@media only screen and (min-width: 480px) and (max-width: 767px) {
    ...
} /* CLOSE WIDE MOBILE LAYOUT */
```

¡Las modificaciones son abundantes y ocupan unas 150 líneas! Así que no vamos a verlas todas.

Las modificaciones principales hacen referencia al tamaño de los contenedores, con un ancho de`<body>` establecido en 320 píxeles y la aplicación de la propiedad `width: 100%` para los principales contenedores.

```
body {
    width: 320px;
    padding: 0;
}
header[role="masthead"] {
    width: 100%;
    margin: 69px 0 20px;
}
nav {
    top: 0;
    left: 0;
    width: 100%;
    ...
}
#content {
    width: 100%;
}
#featured #slides {
    width: 100%;
    font-size: .75em;
}
```

Se han ocultado las imágenes de los vínculos del menú de navegación de la parte superior.

```
nav li#t-tastemakers a, ... {
    background-image: none;
}
```

El logotipo (que se encuentra dentro de un `<h1>`) aparece centrado en el `<header>`:

```
header h1#branding {
    margin: 0 auto;
}
```

El menú de navegación `#nav`, que inicialmente se encontraba debajo del diaporama, ya no es visible.

```
#featured #nav {
    display: none;
}
```

Para las imágenes de los artículos, se ha fijado un ancho máximo del 100%, para que no sean más grandes que su elemento padre.

```
#content article img {
    max-width: 100%;
    height: auto;
}
```

La fuente

La plantilla del gato, **The cat template**, está disponible en la URL:http://freehtml5templates.com/cattemplate-html5-and-css3-template/

En esta dirección tendrá acceso al sitio web de demostración y podrá descargar los archivos fuente.

Template

- For your cats -

Home About Portfolio News Contact

"Beware of people who dislike cats."
– Irish Proverb

Welcome

Douglas Adams: One of the problems of taking things apart and seeing how they work - supposing you're trying to find out how a cat works--you take that cat apart to see how it works, what you've got in your hands is a non-working cat. The cat wasn't a sort of clunky mechanism that was susceptible to our available tools of analysis.

I am watching...

Cats don't like change without their consent. Cats can be cooperative when something feels good, which, to a cat, is the way everything is supposed to feel as much of the time as possible. The cat is a creature of most refined and subtle perceptions naturally.

Read more

I am thinking...

There are people who reshape the world by force or argument, but the cat just lies there, dozing; and the world quietly reshapes itself to suit his comfort and convenience.

Read more

I am resting...

The thing about cats, As you may find, Is that no one knows What they have in mind.

Read more

And purr sleeping..

Cats have always been associated with the Moon. Like the Moon, they come to life at night, escaping from humanity and wandering over housetops with their eyes beaming out through the darkness.

Read more

Fernand Mery quote

With the qualities of cleanliness, affection, patience, dignity, and courage that cats have, how many of us, I ask you, would be capable of becoming cats? God made the cat in order that man might have the pleasure of caressing the lion.

Are cats lazy? Well, more power to them if they are. Which one of us has not entertained the dream of doing just as he likes, when and how he likes, and as much as he likes?

The cat is magical and the bringer of good luck. - Indian

Hello hello

Lorem ipsum dolor sit amet

Lorem ipsum dolor sit amet

Bla bla

Lorem ipsum dolor sit amet

Lorem ipsum dolor sit amet

Purr purr

Lorem ipsum dolor sit amet

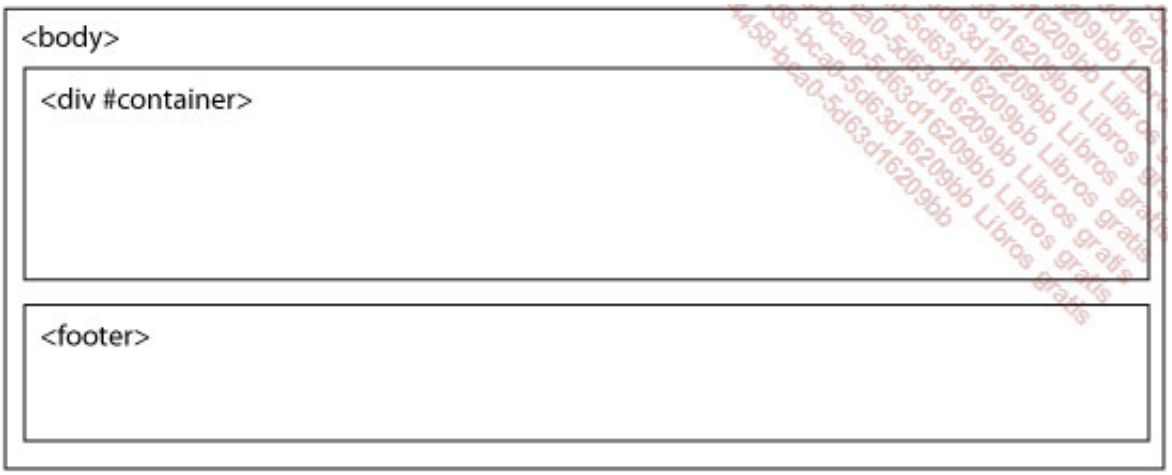
Lorem ipsum dolor sit amet

La estructura HTML5

1. Dos partes

El sitio web está dividido en dos partes principales:

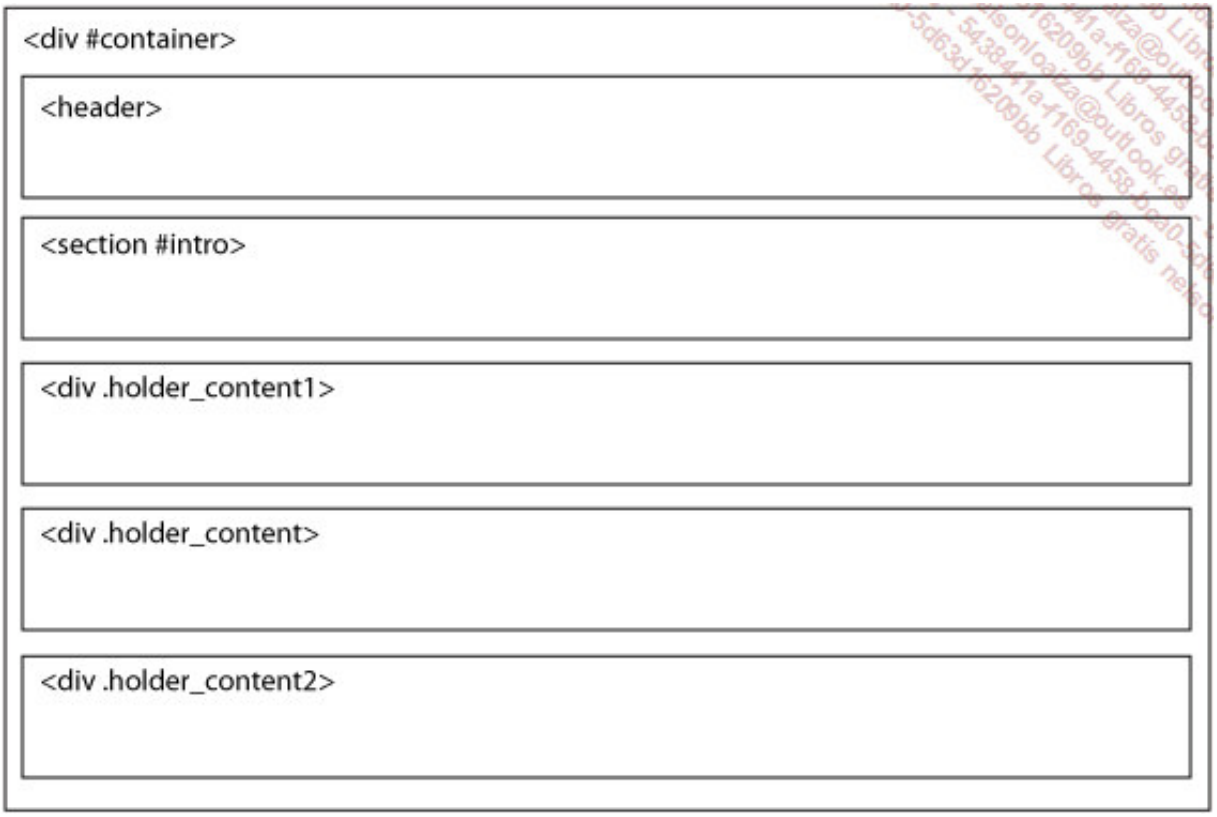
- una caja `<div id="container">`,
- un `<footer>` para el pie de página.



2. El contenedor principal

La caja `<div id="container">` contiene cinco zonas de visualización:

- un `<header>` para el encabezado, con el logotipo y el menú de navegación,
- una sección `<section id="intro">` para el diaporama,
- una caja `<div .holder_content1>` para el mensaje que aparece debajo del diaporama,
- una caja `<div .holder_content>` que muestra los cuatro artículos en columnas,
- una caja `<div .holder_content2>` que muestra un artículo solo, en línea.

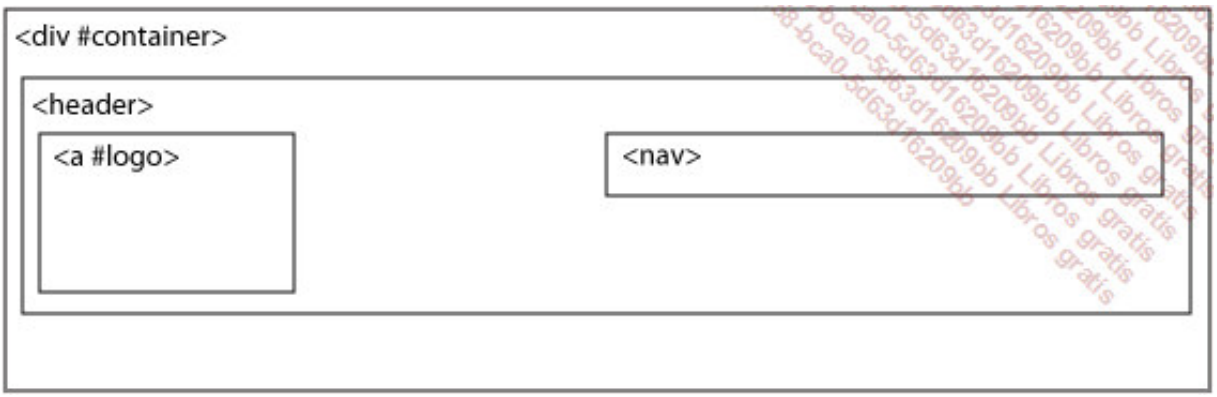


3. El encabezado

Estos son los elementos del encabezado:

El elemento `<header>` contiene:

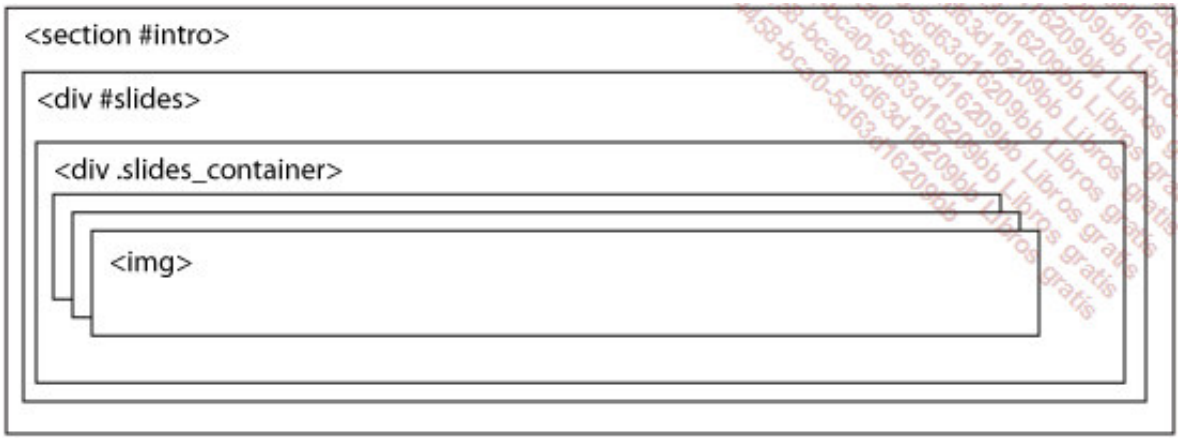
- un vínculo `<a id="logo">`, con una imagen de fondo,
- un elemento `<nav>` para el menú de navegación, que ha sido creado con la habitual lista`<ul>`.



4. El diaporama

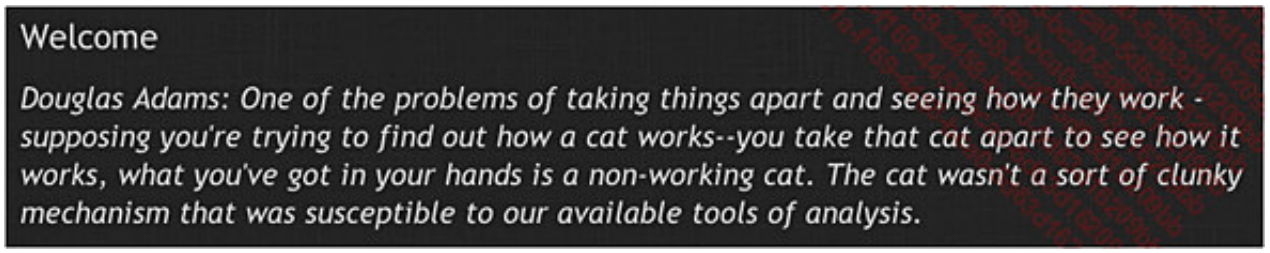
Este es el diaporama:

El diaporama se encuentra en un elemento `<section>` que contiene dos cajas `<div>`, con las imágenes como último hijo.



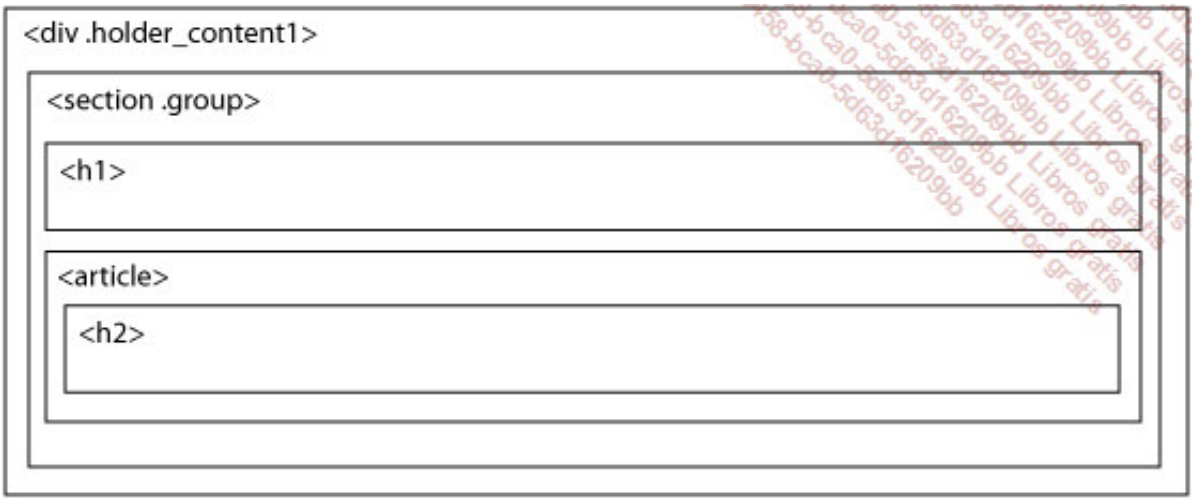
5. El mensaje

Este es el mensaje mostrado:



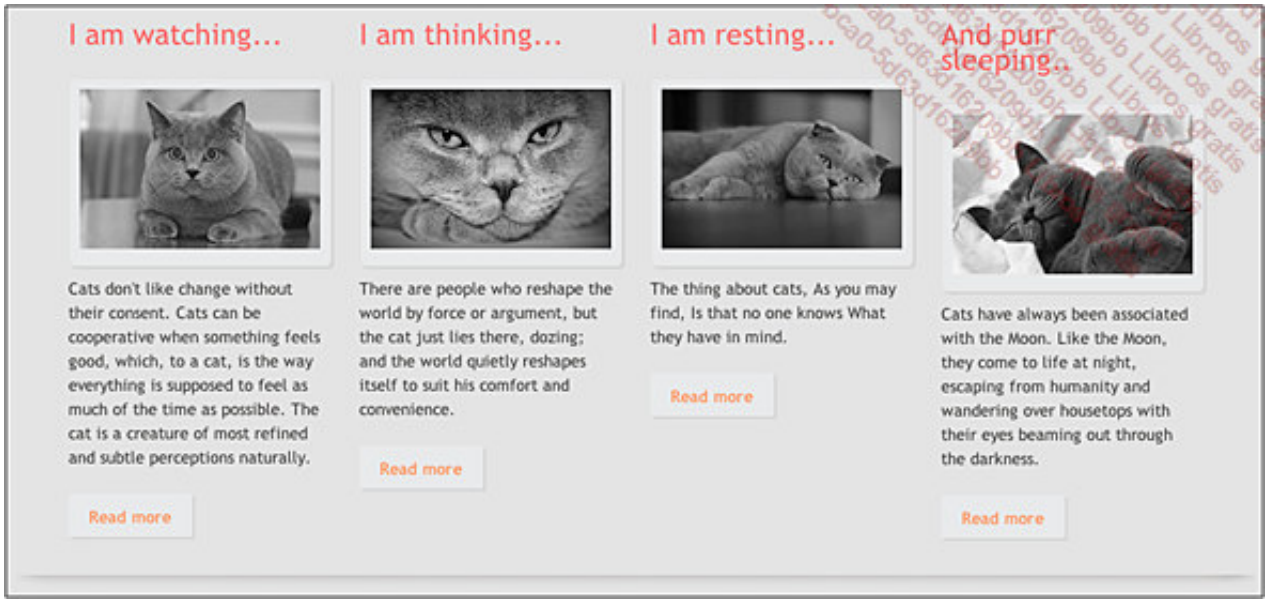
El mensaje que aparece debajo del diaporama se encuentra dentro de una caja `<div class="holder_content1">`, que contiene un elemento `<section class= "groups">`. Este último contiene, a su vez:

- un título `<h1>`,
- un elemento `<article>` que contiene un `<h2>` para el texto.



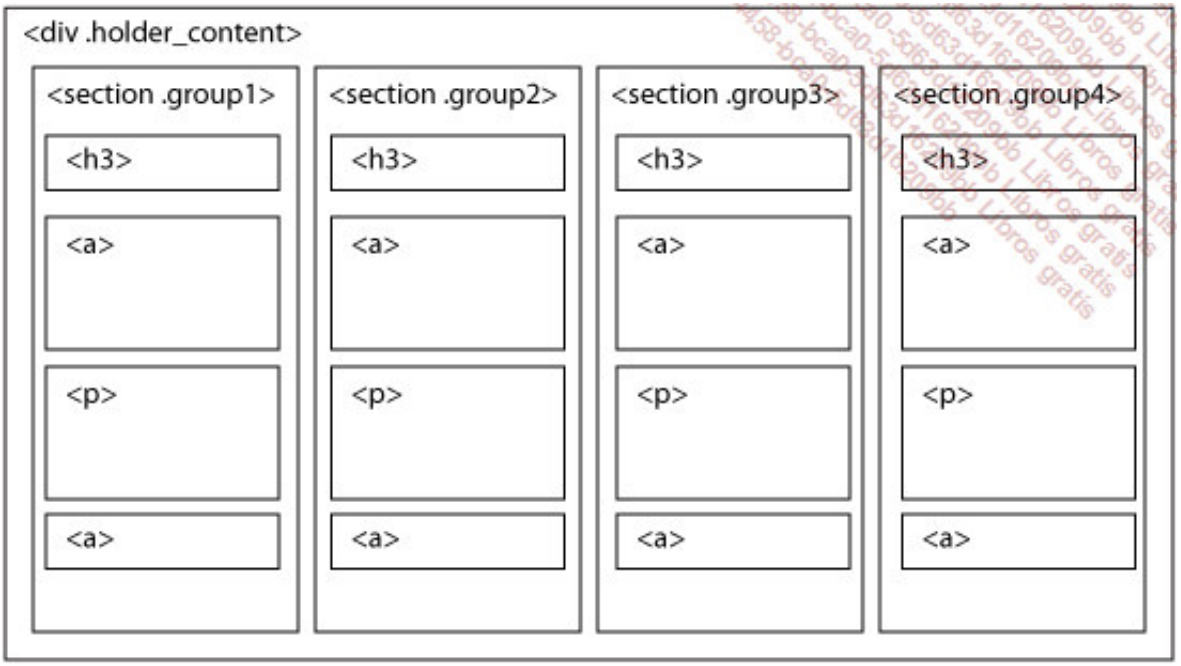
6. Los cuatro artículos

Podemos ver cuatro artículos en columnas:



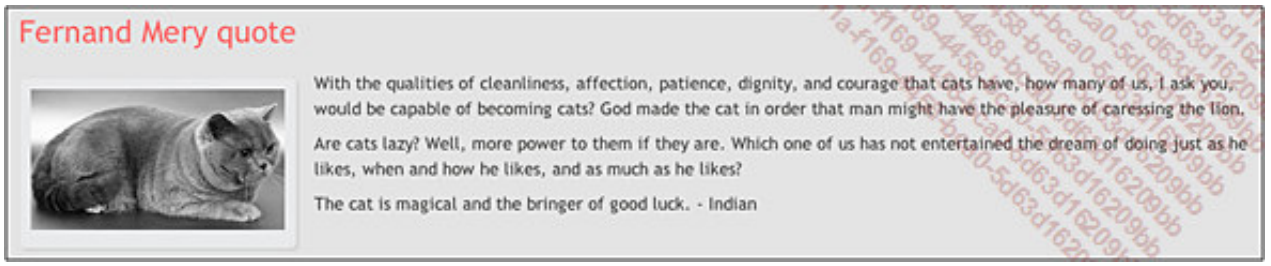
Los cuatro artículos que aparecen verticalmente se encuentran dentro de una caja `<div .holder_content>`. Cada artículo ha sido insertado en un elemento `<section>` con una clase específica. Cada `<section>` contiene:

- un título `<h3>`,
- un vínculo `<a>` con una imagen de fondo,
- un párrafo `<p>` para el texto,
- un vínculo `<a>` para el botón **Read more**.



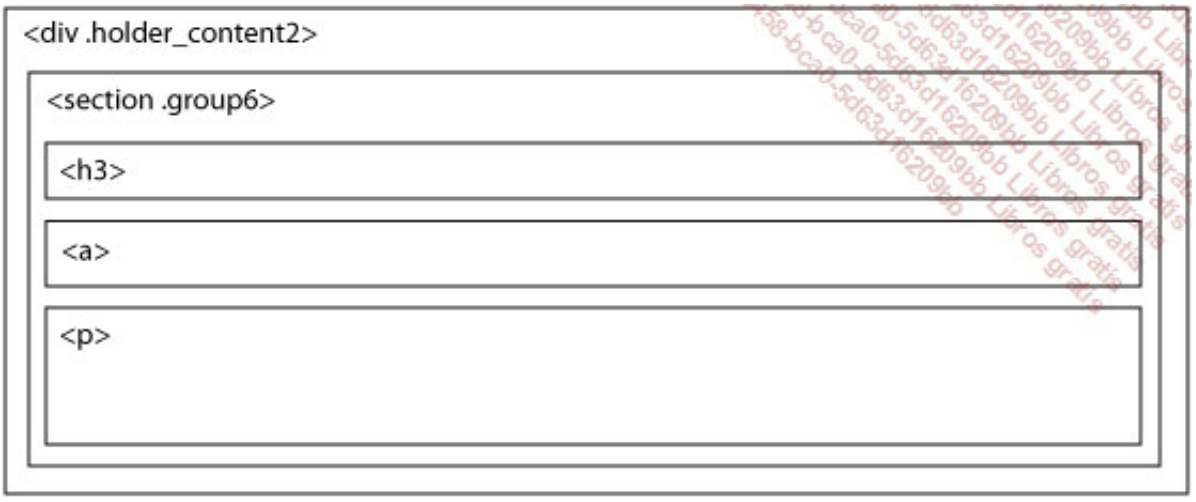
7. El artículo solo

Este es el artículo solo:



Podemos ver a continuación un artículo que aparece solo. Este ha sido insertado en una caja `<div .holder_content2>` que contiene un elemento `<section .group6>`. Este elemento`<section>` contiene, a su vez:

- un título `<h3>`,
- un vínculo `<a>` con una imagen de fondo,
- los párrafos `<p>`.

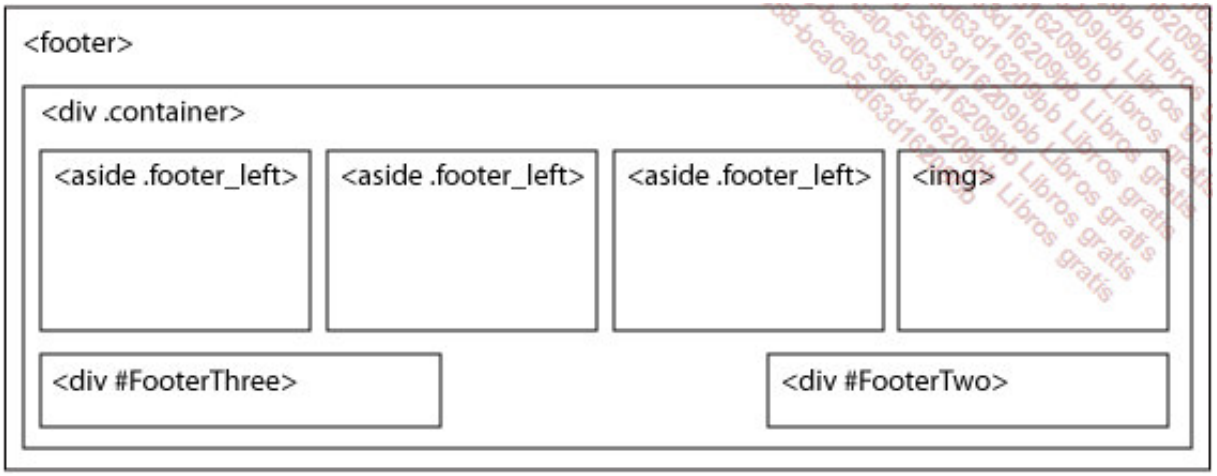


8. El pie de página

Así se visualiza el pie de página:

El pie de página `<footer>` contiene una caja `<div class="container">`. Esta contiene, a su vez:

- tres elementos `<aside class="footer_left">`, para los tres textos del pie de página,
- una imagen `<img>`,
- una caja `<div #FooterThree>` en el lado izquierdo, para mostrar la validación HTML5,
- una caja `<div #FooterTwo>` en el lado derecho, para incluir el copyright.



El diseño CSS3

1. Objetivo

No vamos a describir aquí todo el diseño CSS, que es además bastante clásico, sino que vamos simplemente a aprehender los estilos CSS3.

2. El diaporama

El diaporama se encuentra dentro de un elemento `<section id="intro">` y presenta bordes redondeados.

Este es el estilo CSS:

```
#intro{
    width: 960px;
    position: relative;
    float: left;
    height:300px;
    padding:10px;
    background:#E9EAEB;
    margin-top:52px;
    -moz-border-radius: 5px;
    border-radius: 5px;
}
```

Como podemos ver, se han usado bordes redondeados de 5 píxeles: `border-radius: 5px;`.

3. Las fotografías de los artículos

En los artículos en columnas, las fotos de gatos presentan bordes redondeados y una sombra.

Este es el estilo CSS que se ha utilizado:

```
a.photo_hover3{
    position:relative;
    float: left;
    margin:5px 13px 8px 0;
    padding: 8px;
    -moz-border-radius: 5px;
    border-radius: 5px;
    -moz-box-shadow: 2px 2px #D7D7D7;
    -webkit-box-shadow: 2px 2px #D7D7D7;
}
```

Como podemos ver, se han usado bordes redondeados de 5 píxeles: `border-radius: 5px;` y se ha aplicado una sombra de 2 píxeles de color gris claro: `box-shadow: 2px 2px #D7D7D7;`.

4. Al pasar el cursor sobre las fotografías de los artículos

Al pasar el cursor con el ratón sobre las fotos, podemos ver un efecto de transparencia que se consigue gracias a la propiedad `opacity`:

```
a.photo_hover3:hover {
    border: 1px solid #E1E1E1;
    background-color:white;
    opacity:0.5;
}
```

5. Los botones Read more

Los botones **Read more** son vínculos `<a class="button">`.

Esta clase permite aplicar una sombra: `box-shadow: 2px 2px #D7D7D7;`.

```
.button {
    width: 80px;
    height:16px;
    position:relative;
    margin-top:12px;
    padding:9px;
    background:#E9EAEB;
    display: inline-block;
    color:#FF9966;
    font-weight:bold;
```

```
    cursor: pointer;
    text-align: center;
    font-family:"Trebuchet MS", Arial, Helvetica, sans-serif;
    float:left;
    text-decoration:none;
    -moz-box-shadow: 2px 2px #D7D7D7;
    -webkit-box-shadow: 2px 2px #D7D7D7;
}
```

La fuente

Esta plantilla es interesante porque, además de proponer una estructura HTML5 eficaz, emplea los estilos CSS3 para crear la interfaz de tipo acordeón de la página de inicio.

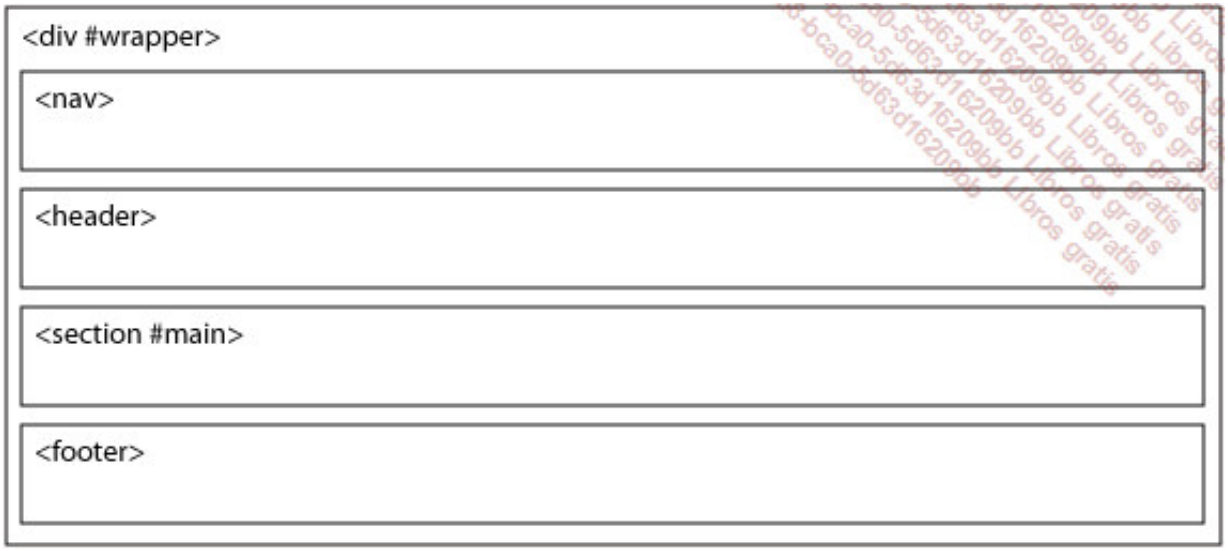
Esta es la URL en la que podrá conseguir los archivos fuente: http://freehtml5templates.com/flipthru-html5-and-css3-template/

La estructura HTML5

1. La caja principal

Esta plantilla utiliza de manera muy directa y acertada los nuevos elementos HTML5. El diseño del sitio web utiliza un contenedor general, una caja `<div id="wrapper">`. Esta caja contiene:

- un menú de navegación en un elemento `<nav>`,
- un encabezado en un elemento `<header>`,
- el contenido central, en un elemento `<section id="main">`,
- y un pie de página, en un elemento `<footer>`.



2. El menú de navegación

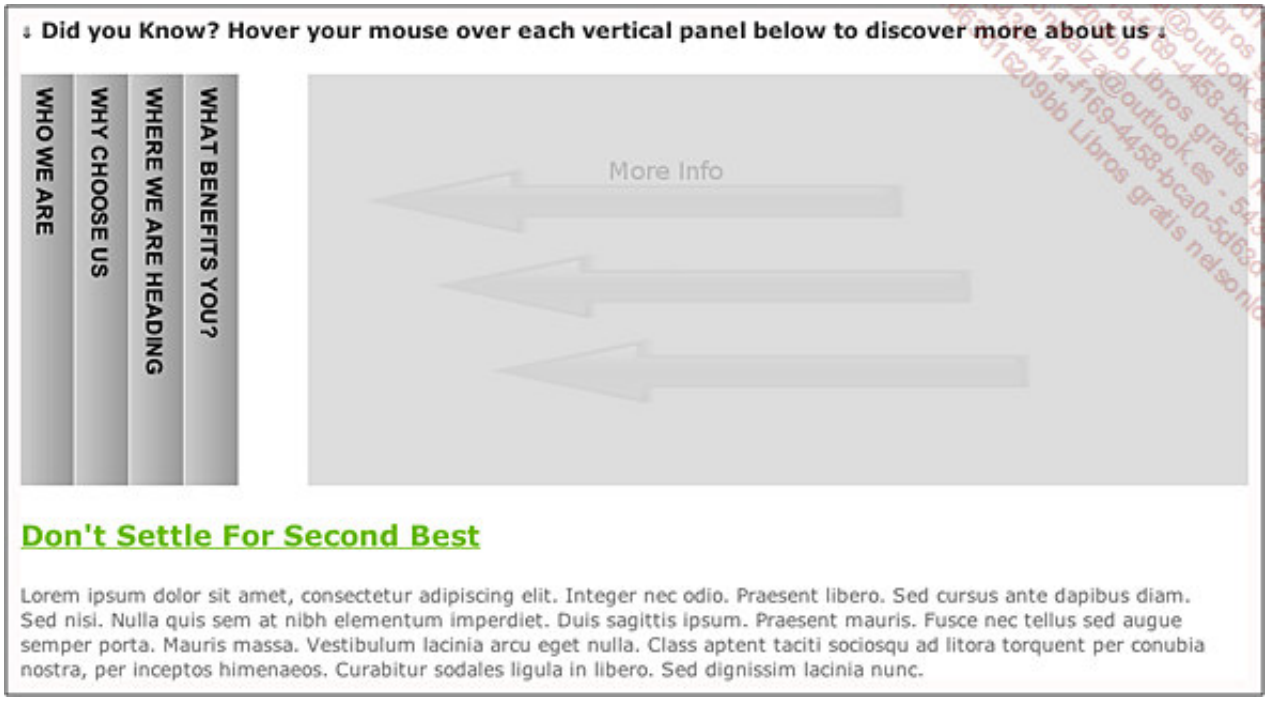
El menú de navegación del elemento `<nav>` ha sido creado con la clásica lista `<ul>`.

3. El encabezado

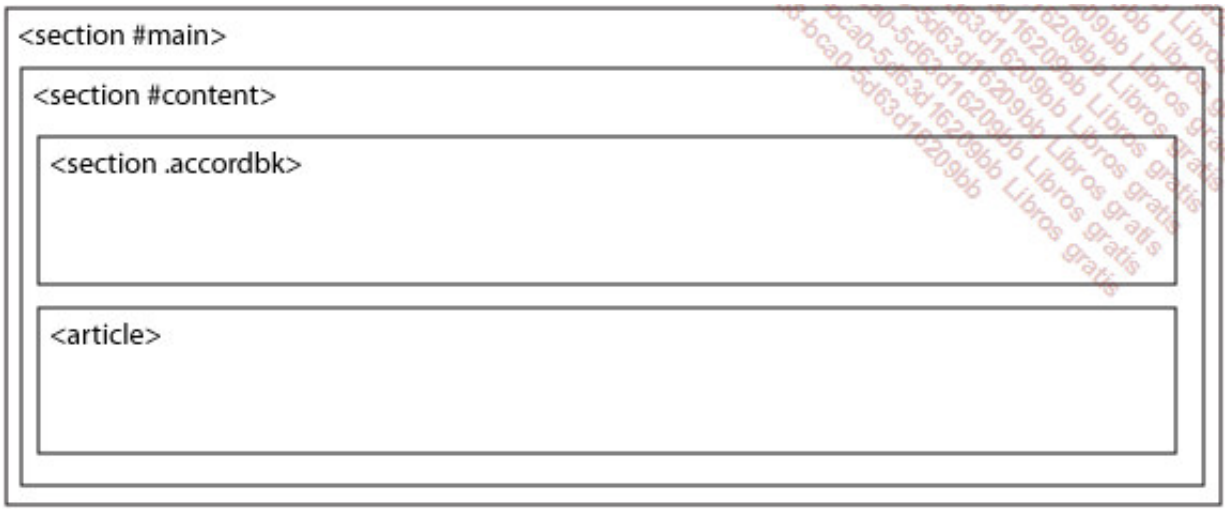
El encabezado `<header>` contiene un título `<h1>` con un vínculo `<a>`, para el título del sitio web, y un título `<h2>`, para el eslogan.

4. La zona central

En la zona central podemos ver el subtítulo, el menú de tipo acordeón y el artículo en portada.



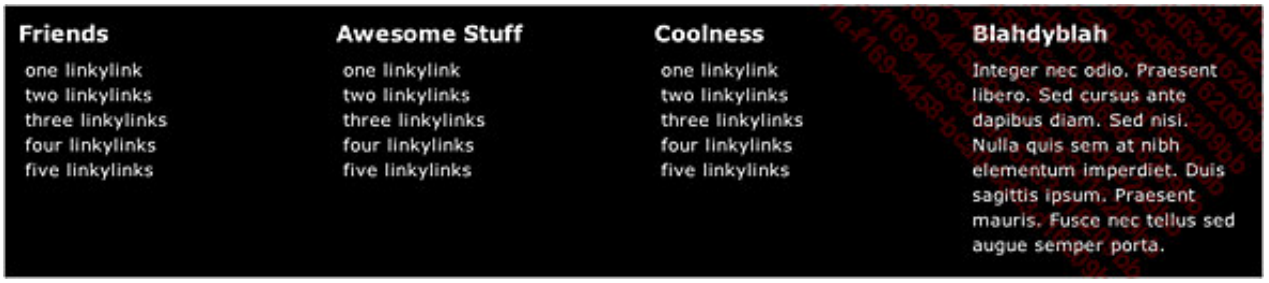
La zona central es un elemento `<section id="main">` que contiene, a su vez, otro`<section id="content">`. Esta caja contiene otro elemento `<section class="accordbk">` para la gestión del acordeón y un elemento `<article>` para presentar el texto (con un `<h2>` y un `<p>`).



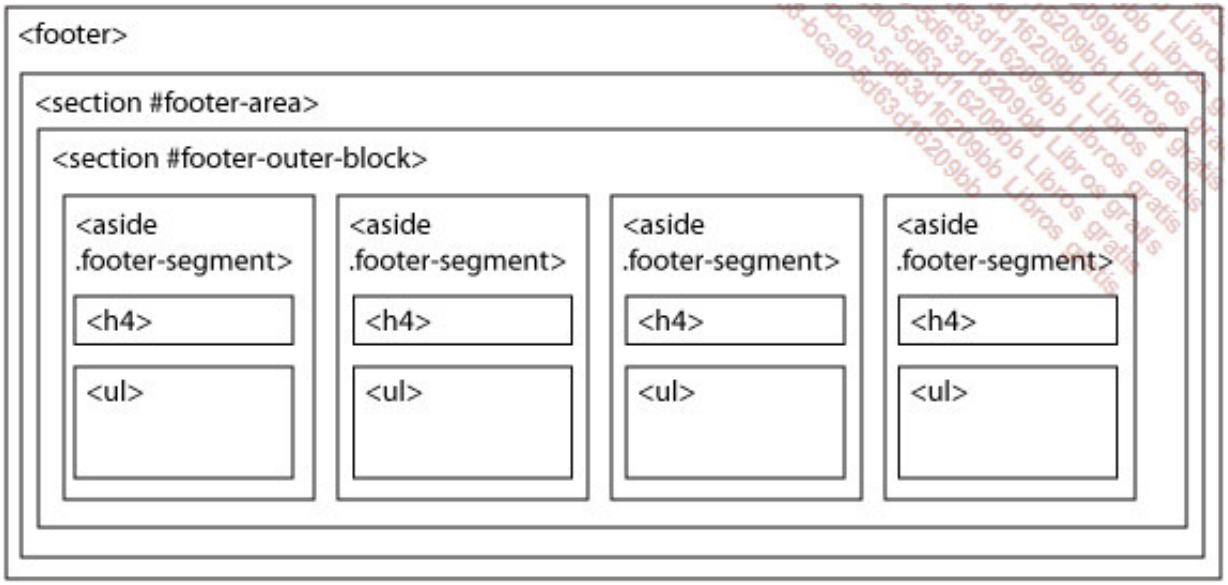
Veremos con más detalle el acordeón en la sección Elemento de interfaz de tipo acordeón.

5. El pie de página

Así se visualiza el pie de página:



El pie de página `<footer>` contiene dos elementos `<section>` imbricados para obtener el diseño deseado. Los cuatro bloques son elementos `<aside>` que contienen un título `<h4>` y una lista`<ul>`.



El diseño CSS3

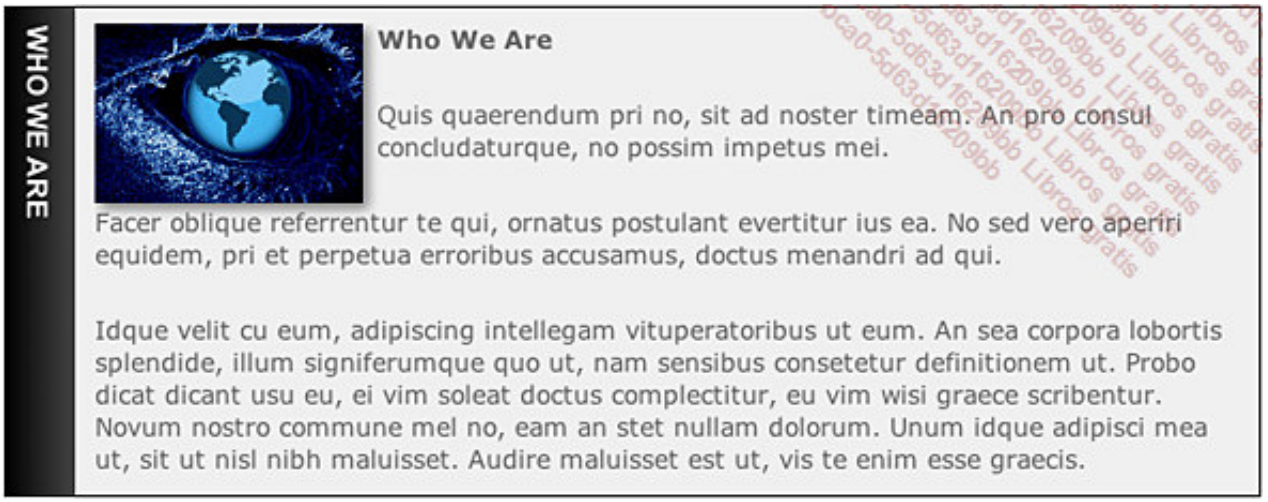
1. El contenedor principal

El contenedor principal es una caja `<div id="wrapper">` a la que se le ha aplicado una sombra`box-shadow: 3px 3px 7px #777`). Este es el estilo CSS:

```
#wrapper {
    width: 940px;
    margin: 0 auto;
    margin-top: 5px;
    margin-bottom: 5px;
    -webkit-box-shadow: 3px 3px 7px #777;
    -moz-box-shadow: 3px 3px 7px #777;
}
```

2. Las imágenes de los artículos

Las imágenes de los artículos del elemento de tipo acordeón presentan una sombra (`box-shadow: 3px 3px 7px #777`).



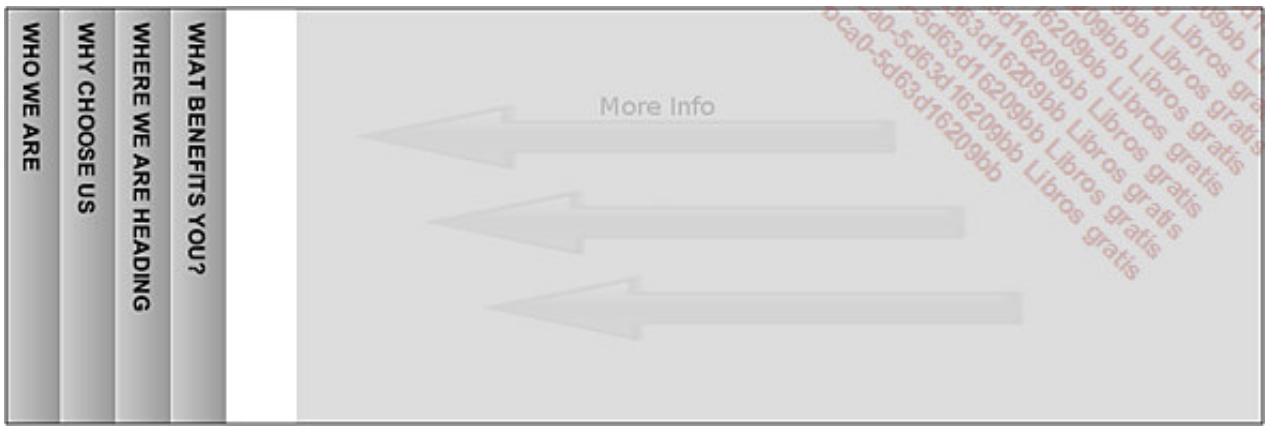

```
article img {
    border: none;
    -webkit-box-shadow: 3px 3px 7px #777;
    -moz-box-shadow: 3px 3px 7px #777;
}
```

El elemento de interfaz de tipo acordeón

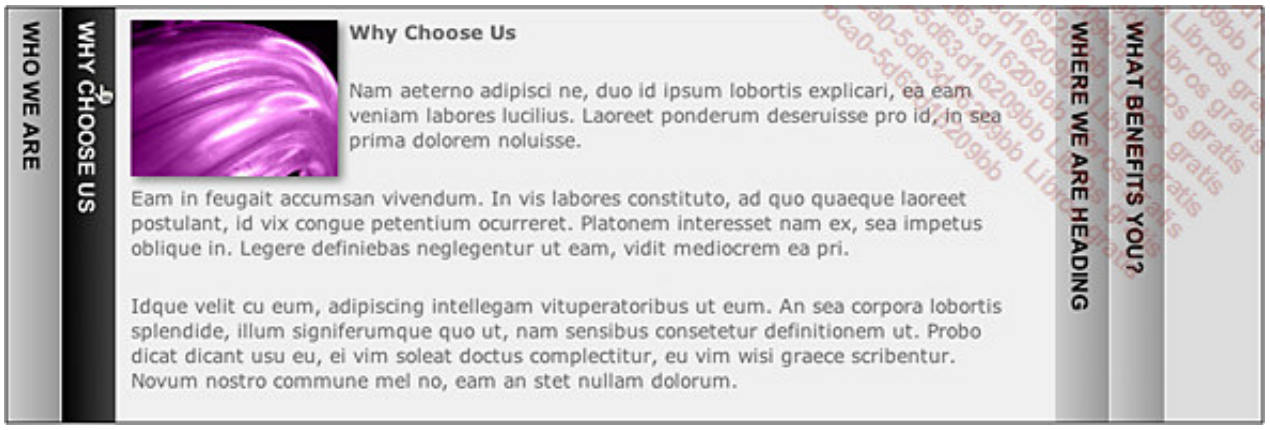
1. El funcionamiento

El menú de tipo acordeón de la zona central funciona, simplemente, al pasar el cursor por encima con el ratón.

Inicialmente se muestra una imagen de fondo:



Al pasar el ratón por encima de los títulos, el acordeón se abre y el artículo en cuestión aparece.

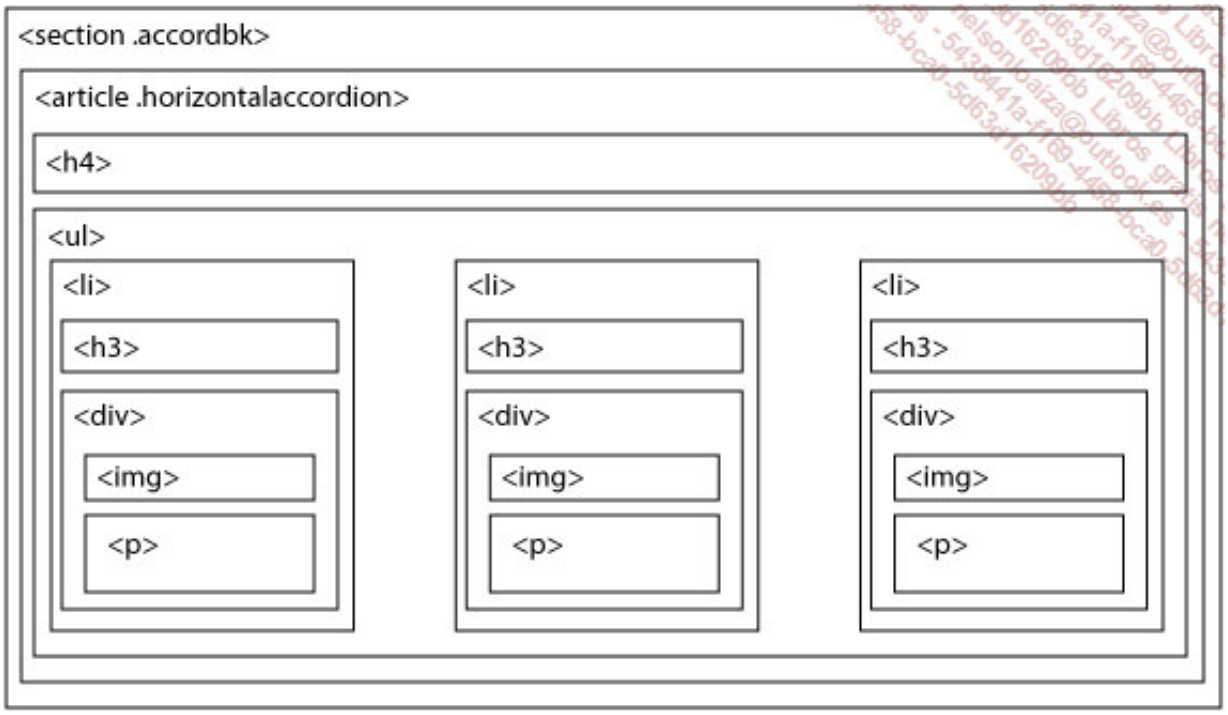


2. La estructura

El acordeón se encuentra dentro de un elemento `<section class="accordbk">`, dentro de un elemento `<article .horizontalaccordion>`. El eslogan ha sido insertado en un elemento `<h4>`.

Los artículos están ordenados en una lista `<ul>`. Cada elemento `<li>` de la lista contiene:

- un título `<h3>`,
- una caja `<div>` que contiene las imágenes `<img>` y los párrafos `<p>` para el texto.



3. El fondo del acordeón

La imagen de fondo del acordeón, con las flechas, es la imagen de fondo del elemento`<section class="accordbk">`:

```
.accordbk {
    background: #fff url(images/accordinfo.gif) bottom right no-repeat;
}
```

4. La lista

La lista `<ul>` presenta un diseño clásico: sin márgenes (`margin: 0`), ni espaciado (`padding: 0`), ni símbolos de enumeración (`list-style:none`) y con una altura fija (`height: 300px`).

```
.horizontalaccordion>ul {
    margin: 0;
    padding: 0;
    list-style: none;
    height: 300px;
}
```

5. Los títulos <h3>

Los títulos `<h3>` de los artículos se muestran verticalmente, con un degradado de fondo. ¡No se trata de una imagen, sino de CSS3!

Se ha usado la propiedad CSS3 `transform()` para aplicar las funciones `rotate()` y`translate()`, así: `transform: rotate(90.0deg) translate(-40px,0px)`. Con la rotación se logra girar el elemento de 90º y la transformación de 40 píxeles lo posiciona correctamente.

Además, los títulos `<h3>` tienen un degradado de fondo: `background: linear-gradient( top, #999999, #cccccc)`.

```
.horizontalaccordion>ul>li>h3 {
    display:block;
    float:left;
    margin: 0;
    padding:10px;
    height:19px;
    width:280px;
    border-left:#f0f0f0 1px solid;
    font-family: Arial, Helvetica, sans-serif;
    text-decoration:none;
    text-transform:uppercase;
    color: #000;
    background:#cccccc;
    white-space:nowrap;
    -moz-transform: rotate(90.0deg) translate(-40px,0px);
    -moz-transform-origin: 0 100%;
    -o-transform: rotate(90.0deg) translate(-40px,0px);
    -o-transform-origin: 0 100%;
    -webkit-transform: rotate(90.0deg) translate(-40px,0px);
    -webkit-transform-origin: 0 100%;
    transform: rotate(90.0deg) translate(-40px,0px);
    transform-origin: 0 100%;
    filter: progid:DXImageTransform.Microsoft.BasicImage
        rotation=1.0)
        progid:DXImageTransform.Microsoft.gradient
        startColorstr=#ff999999, endColorstr=#ffcccccc);
```

```
    -ms-filter: "progid:DXImageTransform.Microsoft.BasicImage
        rotation=1.0)"
     "progid:DXImageTransform.Microsoft.gradient(startColorstr=
#ff999999, endColorstr=#ffcccccc)";
    background: -moz-linear-gradient( top, #999999, #cccccc);
    background: -webkit-gradient(linear, left top, left bottom,
        from(#999999), to(#cccccc));
}
```

6. Los artículos

El contenido de los artículos se encuentra dentro de una caja `<div>`. Inicialmente, esas cajas`<div>` no son visibles, ya que se ha usado la propiedad `display: none`.

Los estilos CSS:

```
.horizontalaccordion>ul>li>div {
    display:none;
    float:left;
    overflow: auto;
    position:relative;
    top:-30px;
    left:50px;
    *top:0px;
    *left:0px;
    margin:0;
    width:640px;
    height:300px;
}
```

7. Las imágenes de los artículos

Los estilos CSS para las imágenes de los artículos no presentan ninguna dificultad:

```
.horizontalaccordion>ul>li>div img {
    float: left;
    margin-right: 8px;
}
```

8. La acción de pasar el cursor

La acción de pasar el cursor por encima con el ratón se lleva a cabo a tres niveles:

Al pasar el cursor por encima de un elemento `<li>` se ocultan los posibles desbordamientos de contenido (`overflow: hidden`) y se fija el ancho (`width: 720px`).

```
.horizontalaccordion>ul>li:hover {
    overflow: hidden;
    width: 720px;
}
```

Al pasar el cursor por encima de los artículos se muestra la caja `<div>` que hasta ahora había permanecido oculta.

```
.horizontalaccordion:hover>ul>li:hover>div {
    display:block;
}
```

Al pasar el cursor por encima de los títulos <h3> se modifica el degradado del fondo, que será entonces más oscuro.

Este es el degradado cuando se pasa el cursor por encima:


```
.horizontalaccordion:hover>ul>li:hover>h3 {
    color:#fff;
    background:#000000;
    background: -moz-linear-gradient( top, #454545, #000000);
    background: -webkit-gradient(linear, left top, left bottom,
       from(#454545), to(#000000));
    filter: progid:DXImageTransform.Microsoft.
       BasicImage(rotation=1.0)
    progid:DXImageTransform.Microsoft.gradient(startColorstr=
#ff454545, endColorstr=#ff000000);
    -ms-filter: "progid:DXImageTransform.Microsoft.
       BasicImage(rotation=1.0)"
    "progid:DXImageTransform.Microsoft.
       gradient(startColorstr=#ff454545, endColorstr=#ff000000)";
}
```

9. La aparición de los artículos

Los artículos aparecerán al pasar el cursor por encima (`:hover`) de los títulos `<h3>`.

```
.horizontalaccordion>ul>li>h3:hover {
    cursor:pointer;
}
```

Al pasar el cursor por encima de los títulos `<h3>` se provoca la transición de los elementos `<li>`. Esta transición se aplica a la propiedad del ancho (`width`), en 0,3 segundos (`0.3s`), con una aceleración (`ease-in-out`): `transition: width 0.3s ease-in-out`.

```
.horizontalaccordion>ul>li {
    display:block;
    overflow: hidden;
    float:left;
    margin: 0;
    padding: 0;
    list-style:none;
    width:40px;
    height: 300px;
    background:#f0f0f0;
    transition: width 0.3s ease-in-out;
    -moz-transition: width 0.3s ease-in-out;
    -webkit-transition: width 0.3s ease-in-out;
    -o-transition: width 0.3s ease-in-out;
}
```

La fuente

El sitio web dConstruct (http://2011.dconstruct.org/), además de presentar una estructura en HTML5, utiliza transiciones interesantes sobre las imágenes de los oradores, incorpora fuentes tipográficas y propone un diseño adaptable. ¡No se puede pedir más!

dConstruct 2011

2nd September — Brighton, England

CONFERENCE WORKSHOPS LOCATION

DESIGNING DIGITAL PRODUCTS

dConstruct 2011 brings together leading thinkers from the fields of interaction design, mobile design and ubiquitous computing to explore how we can bridge the gap between physical and digital product design.

News

5 OCTOBER

dConstruct 2011 Videos

Thanks to our friends at Vzaar, you can now watch **videos of all the talks** from this year's conference. Enjoy!

26 SEPTEMBER

dConstruct 2011 Podcast

We hope you enjoyed dConstruct 2011—we had a blast. If you'd like to re-live the magic, you can now listen to audio recordings of all the talks. Subscribe to **the podcast feed** or cherry-pick the talks you want **on Huffduffer**.

12 JULY

Welcoming more sponsors

We're thrilled to have more super sponsors for this year's dConstruct: **Authentic Jobs**, **Mailchimp**, **Shopify**, and **vzaar**. Welcome aboard!

Sponsors

Executive Sponsors

UBELLY.com

FONT DECK

Media Sponsors

Lanyrd.com

Associate Sponsors

HOEFLER & FRERE-JONES

authentic jobs

MailChimp

vzaar

shopify

theguardian

Proud to be part of the

BRIGHTON DIGITAL FESTIVAL

A season of exhibitions, meet-ups, workshops and outdoor events running throughout September.

brightondigitalfestival.co.uk

Interested in becoming a sponsor?

We have a number of different packages to suit your business needs. **Download our information pack** (1.6Mb PDF) for more information.

HOME PRIVACY POLICY TERMS ACCESSIBILITY CONTACT

Presented by Clearleft, a user experience design consultancy based in Brighton, UK.

We make websites, and in our spare time we like to give something back to the web design community by running dConstruct. It's a grass-roots conference that gathers some of the brightest minds in the industry from around the world, and brings them to our little home by the sea for a cup of tea and a slice of cake.

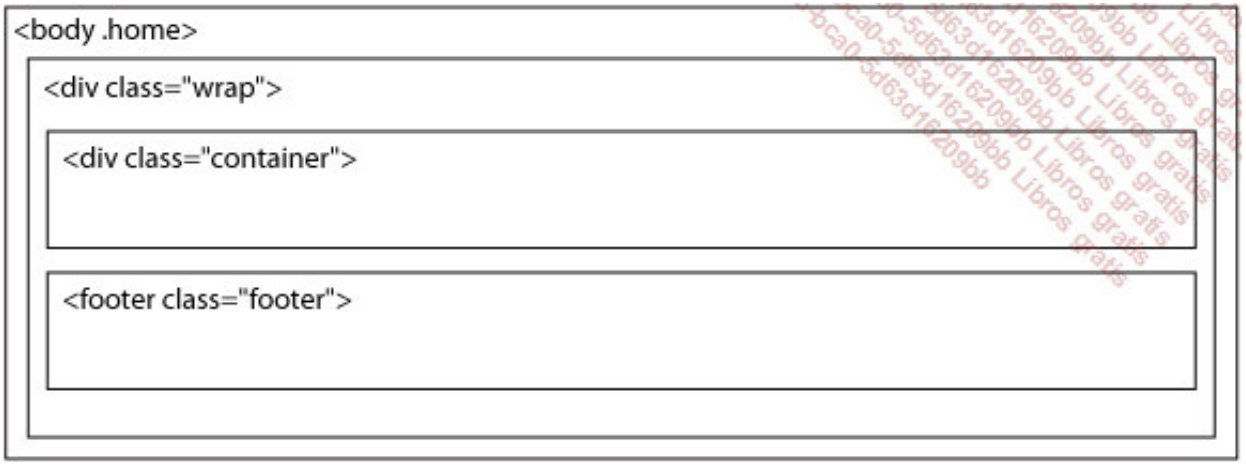
Clearleft

La estructura HTML5

1. La estructura principal

En el `<body class="home">`, encontramos una caja `<div class="wrap">` que contiene:

- una caja `<div class="container">`, para la zona principal de arriba.
- un `<footer class="footer">`, para el pie de página, al final, debajo de la barra de navegación de color negro.



2. El encabezado

Comenzaremos el análisis de este diseño por el encabezado:

En la caja `<div class="container">`, tenemos un `<header class="mast vevent">`que se compone de:

- un título `<h1>`, para el nombre del sitio web, con una fuente tipográfica incorporada y un vínculo `<a>`.
- una caja `<div class="date">`, para mostrar la fecha.
- un elemento `<nav class="main-nav">`, para el menú de navegación principal.

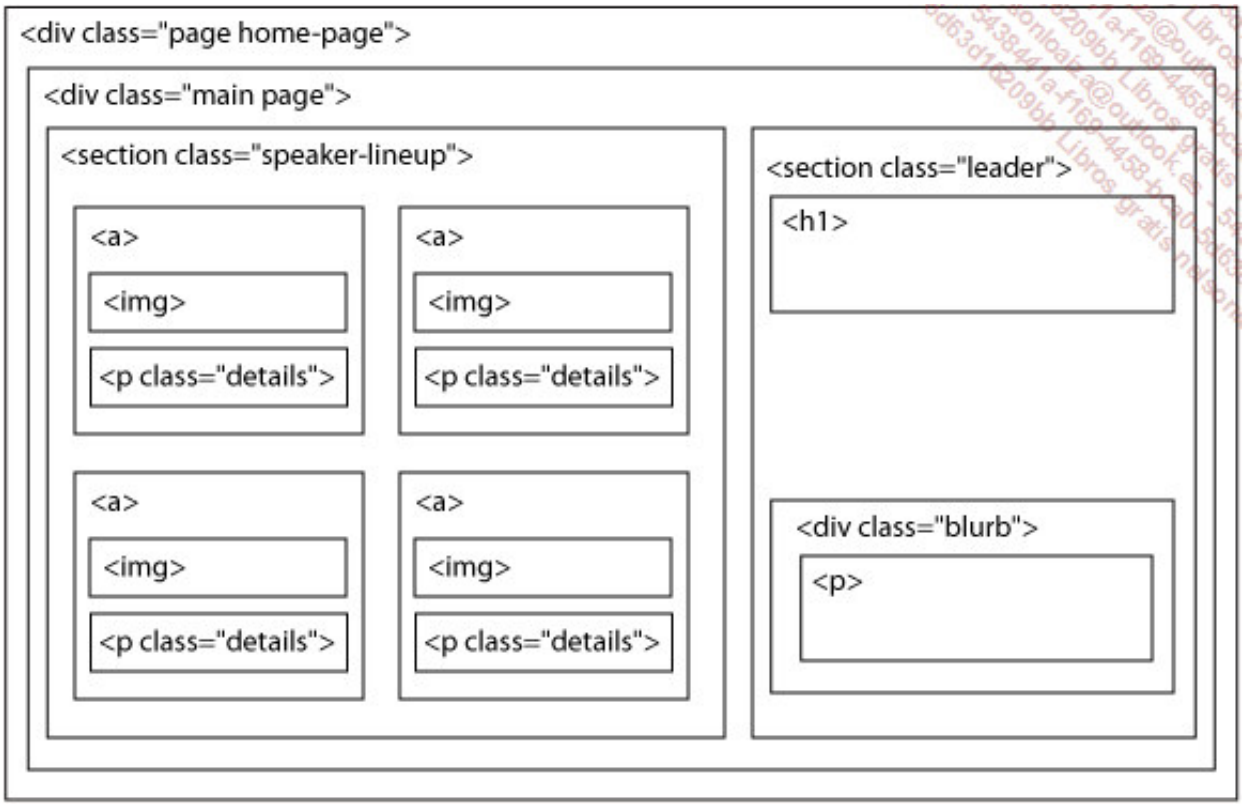
Los artículos están ordenados en una lista `<ol>`.

3. Los oradores

La zona central se visualiza mediante la caja `<div class="container">`. En esta caja encontramos otras dos cajas imbricadas: `<div class="page home-page">` y `<div class="main page">`. Esta última caja contiene las fotografías de los oradores y el eslogan de la derecha.

La caja `<div class="main page">` contiene:

- un elemento `<section class="leader">` que incluye:
  - un `<h1>` para presentar el título con una fuente tipográfica incorporada,
  - una caja `<div class="blurb">` para presentar, en un párrafo `<p>`, el eslogan con una fuente tipográfica incorporada.
- un elemento `<section class="speaker-lineup">` que contiene:
  - un elemento `<a>` para cada orador. Cada vínculo incluye una imagen `<img>` y un párrafo`<p class="details">` para mostrar la información sobre cada orador. Este párrafo inicialmente está oculto.

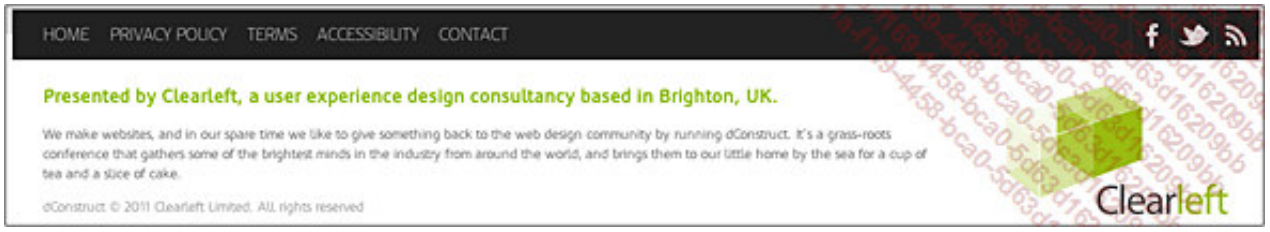


4. Los patrocinadores

La lista de patrocinadores se muestra mediante una aplicación clásica de cajas `<div>` imbricadas.

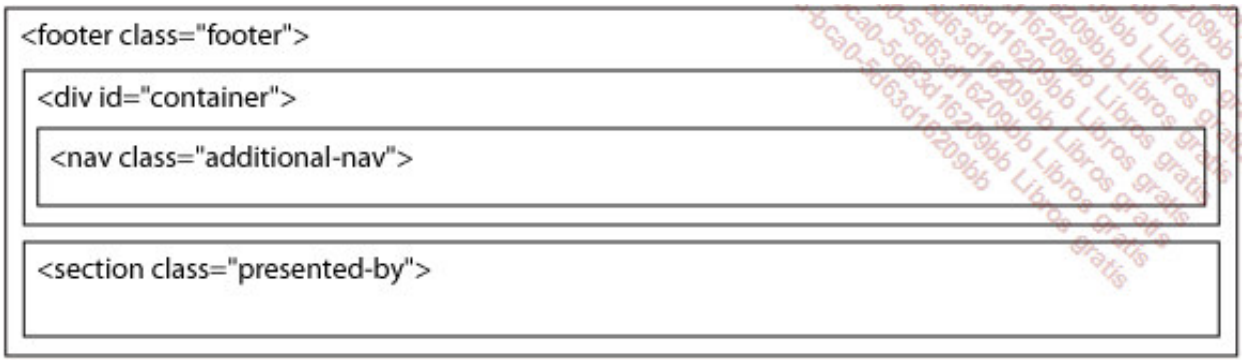
5. El pie de página

El pie de página contiene la barra de navegación secundaria y muestra información adicional.



El pie de página, `<footer class="footer">`, incluye:

- una caja `<div id="container">`, que contiene un elemento `<nav class="additional-nav">` para el menú de navegación. Las dos listas de vínculos (para la navegación y para las redes sociales) son listas `<ol>`.
- una `<section class="presented-by">` para mostrar la información adicional.


<footer class="footer">

<nav class="additional-nav">
<section class="presented-by">

El diseño CSS3

1. Las tipografías incorporadas

El sitio web utiliza varias fuentes tipográficas incorporadas, especialmente para los títulos y los subtítulos.

dConstruct 2011 Podcast

Las reglas para la incorporación de las fuentes se encuentran en el archivo **7047.css**. Veamos, por ejemplo, la incorporación de la primera fuente, sabiendo que el principio será el mismo para las demás fuentes.

Las fuentes están disponibles en los formatos .eot, .woff, Embedded OpenType y OpenType.

```
@font-face {
   font-family: 'Oblik Bold';
   src: url('http://f.fontdeck.com/f/1/
bkl3bWdnWmMAAW56umtJwvfGtJv9w85rBXJ9bDpI/
766OxxQi2WJAONUOwF4RmBRoZY.eot');
   src: url('http://f.fontdeck.com/f/1/
bkl3bWdnWmMAAW56umtJwvfGtJv9w85rBXJ9bDpI/
766OxxQi2WJAONUOwF4RmBRoZY.eot?') format('embedded-opentype'),
   url('http://f.fontdeck.com/f/1/
bkl3bWdnWmMAAW56umtJwvfGtJv9w85rBXJ9bDpI/
766OxxQi2WJAONUOwF4RmBRoZY.woff') format('woff'),
   url('http://f.fontdeck.com/f/1/
bkl3bWdnWmMAAW56umtJwvfGtJv9w85rBXJ9bDpI/
766OxxQi2WJAONUOwF4RmBRoZY.ttf') format('opentype');
   font-weight: bold;
   font-style: normal;
}
```

Las fuentes tipográficas que han sido incorporadas son: Oblik Bold, Oblik Black, Oblik Regular y Oblik Light.

Las fuentes tipográficas aparecen en más de treinta estilos CSS. Veamos solo algunos:

Para los títulos <h1>:

```
h1 {
    font-family: "Oblik Light", "Helvetica Neue", Arial,
Helvetica, sans-serif;
    font-weight: 200;
    font-size: 36px;
    color: #666666;
    clear: both;
    line-height: 0.9em;
}
```

Para el eslogan, en el bloque de la derecha:

```
.blurb p:first-child {
    font-size: 18px;
    line-height: 1.333em;
    margin-bottom: 2em;
     font-family: "Oblik Light", "Helvetica Neue", Arial, Helvetica,
sans-serif;
    font-weight: 200;
    color: #CCC;
}
```

En el menú de navegación principal:

```
.main-nav a {
    display: block;
    text-align: center;
    font-family: "Oblik Bold", "Helvetica Neue", Arial, Helvetica,
sans-serif;
    font-weight: bold;
    color: #555;
    text-shadow: 0 1px 0 rgba(255,255,255,0.75);
    background-image: url("../img/bck/nav.png");
    background-repeat: no-repeat;
    padding-top: 56px;
}
```

En el menú de navegación secundaria del pie de página:

```
.footer .additional-nav a {
    font-family: "Oblik Light", "Helvetica Neue", Arial,
Helvetica, sans-serif;
    font-weight: 200;
    padding-bottom: 0.5em;
}
```

2. Las sombras aplicadas al texto

Algunos textos presentan una sombra discreta, como es el caso de los vínculos del menú de navegación principal.


```
.main-nav a {
    display: block;
    text-align: center;
     font-family: "Oblik Bold", "Helvetica Neue", Arial, Helvetica,
```

```
sans-serif;
    font-weight: bold;
    color: #555;
    text-shadow: 0 1px 0 rgba(255,255,255,0.75);
    background-image: url("../img/bck/nav.png");
    background-repeat: no-repeat;
    padding-top: 56px;
}
```

3. Las sombras de los contenedores

Se han usado los estilos CSS para aplicarle una sombra a la caja `<div class="page home-page">`:

```
.page {
    -webkit-box-shadow: 1px 2px 3px 0px rgba(0,0,0,0.25);
    -moz-box-shadow: 1px 2px 3px 0px rgba(0,0,0,0.25);
    -o-box-shadow: 1px 2px 3px 0px rgba(0,0,0,0.25);
    box-shadow: 1px 2px 3px 0px rgba(0,0,0,0.25);
}
```

Los efectos al pasar el cursor sobre los oradores

1. El funcionamiento

La página de inicio muestra una fotografía de cada orador.

Al pasar el ratón por encima, aparecerá progresivamente un texto de descripción, de abajo hacia arriba:

Ahora podemos ver totalmente la descripción:

2. La estructura

Veamos la estructura de este efecto.

Se ha insertado todo dentro del elemento `<a>` del vínculo. En ese vínculo, encontramos la fotografía del orador en un elemento `<img>`. El texto de descripción se encuentra dentro de un elemento `<p>`que utiliza la clase `details`. El nombre del orador se incluye dentro de un elemento `<strong>`.

Fíjese en la "novedad" en cuanto a la imbricación de los elementos. En HTML 4, los elementos estaban clasificados en dos categorías principales en función de su visualización: los elementos de tipo `block`, que se visualizan uno debajo del otro, y los elementos de tipo `inline`, que se visualizan en fila, uno al lado del otro. En principio, solamente se podían insertar elementos de tipo`block`, dentro de otros elementos de tipo `block`, y los elementos de tipo `inline`, solamente se podían insertar dentro de elementos de tipo `block`. En HTML5, estas nociones han quedado obsoletas. Usted puede insertar elementos unos dentro de otros, como usted quiera, siempre y cuando especifique cómo deberán visualizarse con la propiedad `display`.

En este ejemplo, tenemos un elemento `<a>` que contiene un elemento `<img>` y un elemento `<p>`. Así de sencillo, sin que suponga un problema para la validación.

3. Los estilos iniciales

El nombre del orador utiliza, en especial, una fuente incorporada:

```
.details strong {
    font-family: "Oblik Bold", "Helvetica Neue", Arial, Helvetica,
sans-serif;
    font-weight: bold;
    font-size: 18px;
    display: block;
    color: #87BA00;
    margin-bottom: 0.2em;
}
```

El párrafo completo inicialmente está oculto mediante la propiedad `display`:

```
.details {
    display: none;
}
```

Las propiedades iniciales del párrafo establecen una visualización en bloque (`display: block`), con una posición absoluta (`position: absolute`). El posicionamiento vertical es del 101% (`top: 101%`) respecto a su elemento padre, el vínculo `<a>`. De este modo se consigue colocarlo debajo de la imagen incluida en el vínculo. El posicionamiento izquierdo es del 0 % (`left: 0`) respecto a su elemento padre, el vínculo `<a>`. De este modo se consigue colocarlo correctamente a la izquierda de la imagen incluida en el vínculo. Por último, el ancho (`width: 100%`) y el alto (`height: 100%`) son iguales a los del elemento padre, el vínculo `<a>`.

```
.details {
    display: block;
    position: absolute;
    top: 101%;
    left: 0;
    width: 100%;
    height: 100%;
    ...
}
```

Las demás propiedades hacen referencia al formato y no intervienen en la transición.

4. Al pasar el cursor sobre el vínculo

Cuando se pase el cursor (`:hover`) sobre el vínculo (`a`) se aplicará la transición de la clase `.details` a los párrafos. La propiedad que varía es el posicionamiento vertical, que pasará al 60% (`top: 60%`), como siempre, respecto a su elemento padre `<a>`.

```
a:hover .details {
    top: 60%;
}
```

Esta transición está definida en el selector correspondiente: `.details`. La transición de la nueva posición vertical durará 0,3 segundos (`transition: 0.3s`):

```
.details {
    display: block;
    position: absolute;
    top: 101%;
    left: 0;
    width: 100%;
    height: 100%;
```

```
    ...
    -webkit-transition: 0.3s;
    -moz-transition: 0.3s;
    transition: 0.3s;
}
```

El diseño adaptable

En el diseño de este sitio web se ha previsto que pueda adaptarse en función del tamaño de la pantalla que se utilice para su difusión.

Se incluye una regla para las pantallas que tengan un ancho mínimo de 79 em, es decir: 70 x 16 píxeles = 1120 píxeles.

```
@media all and (min-width: 70em) {
    ...
}
```

Se incluye otra regla para las pantallas que tengan un ancho máximo de 63 em, es decir: 63 x 16 píxeles = 1008 píxeles.

```
@media all and (max-width: 63em) {
    ...
}
```

Se incluye otra regla para las pantallas que tengan un ancho máximo de 60 em, es decir: 60 x 16 píxeles = 960 píxeles.

```
@media all and (max-width: 60em) {
    ...
}
```

Se incluye otra regla para las pantallas que tengan un ancho máximo de 23,75 em, es decir: 23,75 x 16 píxeles = 380 píxeles.

```
@media all and (max-width: 23.75em) {
    ...
}
```

Se incluye una última regla para las pantallas que tengan un ancho máximo de 30 em, es decir: 30 x 16 píxeles = 480 píxeles.

```
@media all and (max-width: 30em) {
    ...
}
```

1. La visualización modificada

Así se visualizará el sitio web con un ancho de unos 500 píxeles aproximadamente.

Las principales modificaciones afectan fundamentalmente al ancho de los elementos.

DESIGNING DIGITAL PRODUCTS

dConstruct 2011 brings together leading thinkers from the fields of interaction design, mobile design and ubiquitous computing to explore how we can bridge the gap between physical and digital product design.

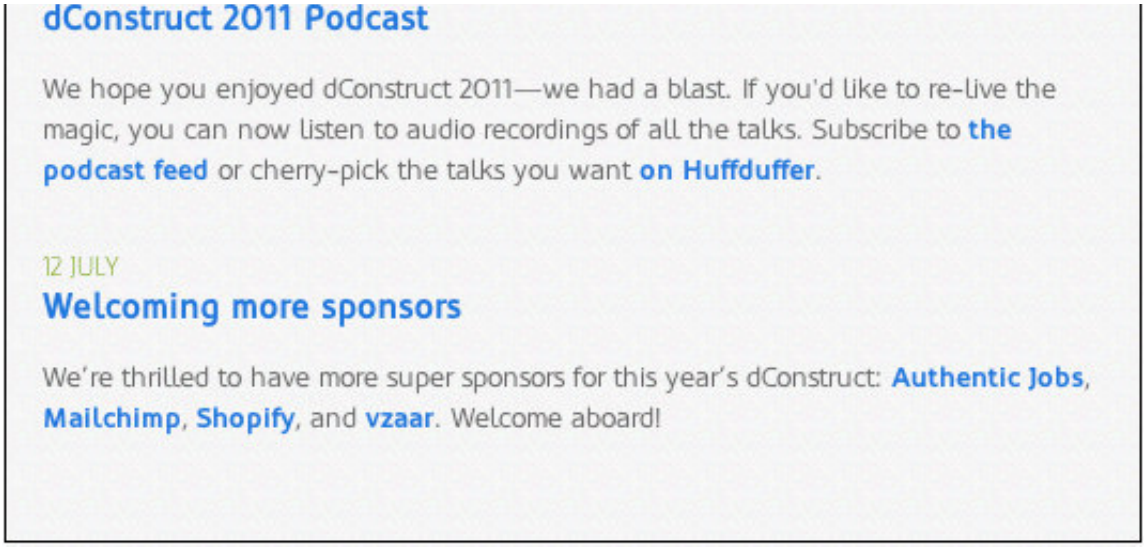
News

5 OCTOBER

dConstruct 2011 Videos

Thanks to our friends at Vzaar, you can now watch **videos of all the talks** from this year's conference. Enjoy!

26 SEPTEMBER



2. El menú de navegación principal

Inicialmente, con el ancho máximo de pantalla, el menú de navegación se compone de vínculos textuales. Al reducirse el tamaño, los vínculos presentarán imágenes.

Este es el estilo utilizado para las pantallas de mayor tamaño: no se visualiza ninguna imagen de fondo (`background: none`):

```
.main-nav a {
    background: none repeat scroll 0 0 transparent;
    font-size: 18px;
    padding: 0.2em 0;
    text-transform: uppercase;
}
```

Este es el estilo utilizado para las pantallas más pequeñas: se indica la imagen que se usará para el fondo (`background-image: url("../img/bck/nav.png")`):

```
.main-nav a {
    display: block;
    text-align: center;
    font-family: "Oblik Bold", "Helvetica Neue", Arial,
Helvetica, sans-serif;
    font-weight: bold;
    color: #555;
    text-shadow: 0 1px 0 rgba(255, 255, 255, 0.75);
    background-image: url("../img/bck/nav.png");
    background-repeat: no-repeat;
    padding-top: 56px;
}
```

3. El título y el eslogan

En una pantalla de menor tamaño, el título y el eslogan se mostrarán sobre las imágenes de los oradores.

Este es el estilo utilizado para la visualización en una gran pantalla, que se aplica al elemento`<section class="leader">`: la caja fluye hacia la derecha (`float: right`):

```
.leader {
    float: right;
    margin-right: 2%;
    padding: 0;
    width: 36.52%;
}
```

Este es el estilo para las pantallas más pequeñas: ya no se produce el desplazamiento y el elemento `<section class="leader">` se coloca en primera posición dentro de su elemento padre, la caja `<div class="main page">`:

```
.leader {
    padding: 0 16px;
}
```

4. Las imágenes de los oradores

En función del ancho de la pantalla, las imágenes de los oradores se organizarán de modo que aparezcan dos, tres o cuatro en cada línea.

Se trata de la regla para las pantallas con un ancho de 30 em como máximo la que impone esta visualización.

La modificación se aplica a los elementos `<a>`, mediante la pseudo-clase `:nth-child()`.

```
@media all and (max-width: 30em) {
    .speaker-lineup a:nth-child(1n+0) {
        width: 50%;
    }
    .speaker-lineup a:nth-child(1n+3) {
        width: 33.3333%;
    }
    .speaker-lineup a:nth-child(1n+6) {
        width: 25%;
    }
}
```

El primer selector, `a:nth-child(1n+0)`, se aplica a las dos primeras imágenes:

- 1n+0 : 1x0+0=0, no hay hijos, no hay imágenes.
- 1n+0 : 1x1+0=1, primer hijo, primera imagen.
- 1n+0 : 1x2+0=2, segundo hijo, segunda imagen.

El tamaño de las dos primeras imágenes será de la mitad del de su elemento padre: `width: 50%`.

El segundo selector, `a:nth-child(1n+3)`, se aplicará a las tres imágenes siguientes:

- 1n+3 : 1x0+3=3, tercer hijo, tercera imagen.

- 1n+3 : 1x1+3=4, cuarto hijo, cuarta imagen.
- 1n+3 : 1x2+3=5, quinto hijo, quinta imagen.

El tamaño de las tres imágenes siguientes será igual a la tercera parte del de su elemento padre:`width: 33,3333%`.

El tercer selector, `a:nth-child(1n+6)`, se aplicará a las imágenes siguientes:

- 1n+6 : 1x0+6=6, sexto hijo, sexta imagen.
- 1n+6 : 1x1+6=7, séptimo hijo, séptima imagen.
- 1n+6 : 1x2+6=8, octavo hijo, octava imagen.
- 1n+6 : 1x3+6=9, noveno hijo, novena imagen.
- ...

El tamaño de las tres imágenes siguientes será igual a la cuarta parte del de su elemento padre:`width: 25%`.